Confidentiel jusqu'au 1er juin 2015

UNIVERSITÉ D'ARTOIS - CRIL CNRS UMR 8188

ONYME SARL

École doctorale : Sciences Pour l'Ingénieur Université Lille Nord-de-France nº 72

Unité de recherche : Centre de Recherche en Informatique de Lens

Thèse présentée par : TROUVILLIEZ Benoît

le 13 mai 2013

Pour obtenir le grade de docteur de l'Université d'Artois

Spécialité : Informatique

Similarités de données textuelles pour l'apprentissage de textes courts d'opinions et la recherche de produits

Thèse dirigée par :

MARQUIS	Pierre	Directeur de thèse
DUBOIS	Vincent	Co-encadrant

Rapporteurs et jury :

SEDES	Florence	Rapporteur
TOUSSAINT	Yannick	Rapporteur
COLLIN	Olivier	Examinateur
KORICHE	Frederic	Examinateur
MARQUIS	Pierre	Directeur de thèse
DUBOIS	Vincent	Co-encadrant
SERNICLAY	Antoine	Invité
VIBES	Thibaud	Invité

Confidential until Monday 1st June, 2015

UNIVERSITÉ D'ARTOIS - CRIL CNRS UMR 8188

ONYME SARL

Doctoral school: Sciences Pour l'Ingénieur Université Lille Nord-de-France nº 72

University department: Centre de Recherche en Informatique de Lens

Thesis defended by: TROUVILLIEZ Benoît

on Monday 13th May, 2013

Submitted in fulfillment of the requirements for the degree of Doctor of Philosophy from Université d'Artois

Speciality: Computer Science

Textual data similarities for learning short opinion texts and retrieving products

Ph. D. thesis supervised by:

MARQUIS	Pierre	Ph.D. supervisor
DUBOIS	Vincent	Co-supervisor

Recorders and dissertation committee:

SEDES	Florence	Ph.D. referee
TOUSSAINT	Yannick	Ph.D. referee
COLLIN	Olivier	Member
KORICHE	Frederic	Member
MARQUIS	Pierre	Ph.D. supervisor
DUBOIS	Vincent	Co-supervisor
SERNICLAY	Antoine	Guest
VIBES	Thibaud	Guest

Mots clés : Traitement Automatique des Langues, Représentation de textes, Correction orthographique, Modèle colorimétrique, Analyse syntaxique, Analyse lexicale, Analyse morphologique, *Synset*, Apprentissage artificiel, Similarité, Dissimilarité, Apprentissage de textes courts d'opinions, Recherche de produits, Recherche de couleurs

Keywords: Natural Language Processing, Text representation, Spell checking, Color model, Syntaxic analysis, Lexical analysis, Morphological analysis, Synset, Machine learning, Similarity, Dissimilarity, Learning short opinion texts, Product retrieval, Color search

Cette thèse a été préparée dans le cadre d'une convention CIFRE

conjointement au

Centre de Recherche en Informatique de Lens
CRIL CNRS UMR 8188 - Université d'Artois
Rue Jean Souvraz
SP 18 - F 62307 Lens Cedex
☎ 03 21 79 17 23 FAX 03 21 79 17 70
✉ gestion@cril.univ-artois.fr
Site http://www.cril.univ-artois.fr/

et à

Onyme SARL
Parc Euratechnologies
165 avenue de Bretagne
59000 Lille
☎ 03 20 42 88 32 FAX 03 20 00 69 38
✉ contact@onyme.com
Site http://www.onyme.com/

Je dédie ce travail

à l'âme d'enseignant de chacun
à celles de mes professeurs
de tous cycles et de toutes formations
sans lesquelles ce travail n'aurait pu être achevé

à ma soeur
à ma mère
à ma grand-mère
au reste de ma famille
et à l'ensemble de mes connaissances
pour leur présence et leur soutien
lors de la rédaction de cet ouvrage

« Le billet d'un dollar que le client reçoit des guichetiers dans quatre banques différentes est le même. Ce qui est différent, c'est les guichetiers »

Marcus STANLEY

« Les bons conseillers ne manquent pas de clients »

William SHAKESPEARE

« Parler est un besoin, écouter est un art »

Johann Wolfgang von GOETHE

« On se souvient de la qualité bien plus longtemps que du prix »

Guccio GUCCI

« Manier savamment une langue, c'est pratiquer une espèce de sorcellerie évocatoire »

Charles BAUDELAIRE

« Le langage est source de malentendus »

Antoine DE SAINT-EXUPÉRY

« The essential problem of NLP evaluation is that the evaluator has to advance between Scylla and Charybdis »

Karen SPARCK JONES

Remerciements

Ce travail a été élaboré sous la direction de Pierre MARQUIS, à qui je tiens à exprimer toute ma gratitude d'avoir accepté d'être le directeur de mes travaux de thèse.

Mes recherches n'auraient pas été complètes sans l'aide de mon encadrant Vincent DUBOIS, que je remercie pour ses conseils éclairés tant dans la rédaction de cette thèse que dans la conduite de mes travaux.

Je tiens également à remercier messieurs Thibaud VIBES et Antoine SERNICLAY, responsables de la société Onyme SARL, pour m'avoir confié la conduite des travaux qui font l'objet de cette thèse. Je les remercie également pour leur soutien et leurs conseils éclairés de directeurs d'entreprise.

Les trois années qui ont été nécessaires à l'élaboration de cette thèse n'auraient pas été les mêmes sans la présence de mes collègues de travail tant à l'entreprise qu'au laboratoire. Je tiens donc à tous les remercier pour les moments que nous avons partagés ensemble.

Je remercie enfin l'ensemble des membres de ma famille qui m'ont soutenu durant la rédaction de cette thèse. Votre présence à mes côtés fut un réconfort dans les moments difficiles. Je remercie particulièrement ma grand-mère pour le temps qu'elle a consacré à la relecture de cette thèse.

Résumé

Cette thèse porte sur l'établissement de similarités de données textuelles dans le domaine de la gestion de la relation client. Elle se décline en deux parties :

– l'analyse automatique de messages courts en réponse à des questionnaires de satisfaction ;

– la recherche de produits à partir de l'énonciation de critères au sein d'une conversation écrite mettant en jeu un humain et un programme agent.

La première partie a pour objectif la production d'informations statistiques structurées extraites des réponses aux questions. Les idées exprimées dans les réponses sont identifiées, organisées selon une taxonomie et quantifiées.

La seconde partie vise à transcrire les critères de recherche de produits en requêtes compréhensibles par un système de gestion de bases de données. Les critères étudiés vont de critères relativement simples comme la matière du produit jusqu'à des critères plus complexes comme le prix ou la couleur.

Les deux parties se rejoignent sur la problématique d'établissement de similarités entre données textuelles par des techniques de Traitement Automatique des Langues (TAL). Les principales difficultés à surmonter sont liées aux caractéristiques des textes, rédigés en langage naturel, courts, et comportant fréquemment des fautes d'orthographe ou des négations. L'établissement de similarités sémantiques entre mots (synonymie, antonymie, etc) et l'établissement de relations syntaxiques entre syntagmes (conjonction, opposition, etc) sont également des problématiques abordées. Nous étudions également dans cette thèse des méthodes de regroupements et de classification automatique de textes afin d'analyser les réponses aux questionnaires de satisfaction.

Abstract

This Ph.D. thesis is about the establishment of textual data similarities in the client relation domain. Two subjects are mainly considered :

– the automatic analysis of short messages in response of satisfaction surveys ;

– the search of products given same criteria expressed in natural language by a human through a conversation with a program.

The first subject concerns the statistical informations from the surveys answers. The ideas recognized in the answers are identified, organized according to a taxonomy and quantified.

The second subject concerns the transcription of some criteria over products into queries to be interpreted by a database management system. The number of criteria under consideration is wide, from simplest criteria like material or brand, until most complex criteria like color or price.

The two subjects meet on the problem of establishing textual data similarities thanks to *Natural Language Processing* (NLP) techniques. The main difficulties come from the fact that the texts to be processed, written in natural language, are short ones and with lots of spell checking errors and negations. Establishment of semantic similarities between words (synonymy, antonymy, ...) and syntactic relations between syntagms (conjunction, opposition, ...) are other issues considered in our work. We also study in this Ph. D. thesis automatic clustering and classification methods in order to analyse answers to satisfaction surveys.

Table des matières

Liste des tableaux

Table des figures

Liste des algorithmes

Introduction

Cette thèse porte sur l'apport du TAL dans la gestion de la relation client. Nous commençons par introduire la relation client et les promesses de l'outil informatique et du TAL dans cette relation. Nous présentons ensuite nos contributions.

L'informatique comme outil d'aide à la relation client

La relation client correspond à l'ensemble des interactions existantes entre un vendeur et ses clients dans le cadre d'une vente.

Les notions de vendeurs et de vente sont ici larges. Les enseignes, les magasins et leurs employés, ou même le site Internet peuvent être perçus comme les vendeurs. Les biens vendus correspondent aussi bien des objets physiques (par exemple, un réseau de magasins) qu'à des services (par exemple, un réseau bancaire). Cette relation a une dimension nationale, voire mondiale selon l'importance des enseignes ou des sociétés. Avec l'Internet, nous constatons même la création d'enseignes et sociétés uniquement axées sur la relation client à distance telles que Mister Good Deal, Cdiscount, Monabanq ou Fortuneo. La relation client devient dès lors difficile à cause de la très grande quantité d'informations émanant des millions de clients à gérer et de la distance séparant le vendeur et ses clients.

L'informatique joue alors le rôle de mise en relation entre le vendeur et ses clients. Les attentes du vendeur et des clients envers lui sont nombreuses. Les clients veulent s'exprimer que ce soit en matière de souhait de produits ou de services à acheter ou consommer ou en matière de souhait dans l'évolution de la relation client qu'entretient le vendeur avec eux. L'outil informatique ne doit pas brider trop fortement l'expression des attentes du client.

Le vendeur veut comprendre les souhaits des clients, de manière simple, compréhensible et synthétique afin d'y répondre au mieux en y consacrant un temps raisonnable. Le vendeur peut ainsi proposer les articles ou services qui conviennent ou faire évoluer la relation qu'il entretient avec ses clients de manière ciblée, cela dans le but d'augmenter la satisfaction de ses clients et conclure des ventes.

Dans un contexte où la quantité d'informations transitant entre les acteurs de la relation croît rapidement, l'enjeu de l'outil informatique est de garantir la qualité et la rapidité de la compréhension et de la résolution des attentes des clients.

Le TAL comme outil de traitement des textes libres

Chaque client a besoin de pouvoir exprimer précisément et le plus complètement possible ses attentes. L'outil informatique par le biais d'Internet est un support de communication efficace mais il ne permet pas naturellement un libre échange à cause de l'incapacité de la machine à comprendre et donc synthétiser le besoin du client.

Pour contourner ce problème, les vendeurs ont souvent recours à des questionnaires de satisfaction à choix multiples, des moteurs de recherche à facettes ou même encore des menus ne laissant pas le choix au client mais permettant le traitement informatique des informations de manière simple.

L'enjeu du traitement automatique des langues est ici de rétablir la possibilité pour le client de s'exprimer librement, avec ses propres mots, tout en garantissant la même capacité de traitement par le vendeur. Cet obstacle majeur est sans doute responsable de l'absence d'une telle démarche dans de nombreuses applications mettant en jeu la relation client.

Dans cette thèse, nous nous intéressons particulièrement à deux cas :

– L'analyse de la satisfaction client au travers de sondages d'opinions effectués dans le cadre d'une relation client. Les sondages sont réalisés suite à un achat dans un magasin, un achat sur Internet, un entretien avec un conseiller bancaire, ...

– La recherche du ou des produits correspondants aux critères du client. Les recherches portent sur les caractéristiques des produits vendus comme la matière, le prix, ... Nous étudions particulièrement le cas de la recherche de couleurs.

Problématiques abordées dans la thèse

Nous présentons à présent les problématiques abordées dans cette thèse plus en détails.

L'analyse de la satisfaction client

L'analyse s'effectue dans le cadre de réponses à des questionnaires de satisfaction. Le but est de construire une synthèse des idées exprimées par les clients sur la relation client. Ces idées peuvent donc porter sur différents thèmes comme les prix, la livraison, la politique commerciale, le personnel, ...

Nous étudions principalement deux cas :

– la synthèse doit être construite depuis les réponses aux questionnaires.

– la synthèse est déjà construite et les réponses considérées doivent être attribuées à chaque idée apparaissant dans celle-ci.

Nous étudions ainsi l'apport des méthodes d'apprentissage non supervisés dans le premier cas et supervisés dans le second.

Nous étudions en outre certaines difficultés que sont :

– Le nombre important de messages comportant plusieurs idées. Bien que nous traitons des verbatims relativement courts, ceux-ci comportent souvent plusieurs idées (par exemple, « la qualité des vêtements et les prix pas chers » comporte deux idées). Pire, selon la précision des idées de la synthèse à générer, un même verbatim peut comporter une ou plusieurs idées (par exemple, « plus de choix sur les chaussures et les pulls » comporte l'unique idée « plus de choix » ou les deux idées « plus de choix de chaussures » et « plus de choix de pulls »). Cela représente une difficulté dans l'emploi des algorithmes d'apprentissage

souvent conçus pour traiter uniquement des messages comportant une seule idée. Nous proposons des techniques d'adaptation d'algorithmes standard aux messages comportant plusieurs idées. En outre, nous étudions des approches visant à déterminer de manière statistique si un message comporte ou non plusieurs idées.

– Les fautes d'orthographe et de syntaxe sont fréquentes. Cela nous conduit à privilégier des méthodes d'analyse robustes en contexte difficile. Nous étudions également diverses approches de correction des plus naïves aux plus élaborées : approche par racinisation, approches par édition de chaîne de caractères, approches phonétiques et consonantiques.

– Certains messages sont ambigus dans les idées qu'ils expriment (par exemple, « les prix des produits »). Le client est-il content ou mécontent des prix des produits ? Nous montrons comment la prise en compte d'une note dans le calcul de similarité entre messages peut permettre d'offrir une désambiguïsation de ces cas.

La recherche de produits

Nous devons transcrire des messages textuels exprimant des critères de recherche de produits en une requête compréhensible par un système de gestion de base de données. Ces messages sont issus d'une conversation avec un agent virtuel en ligne sur des sites de e-commerces. Ce projet est réalisé dans le cadre d'une prestation de recherche avec des sociétés tiers.

Plusieurs difficultés doivent être gérées :

– Les fautes d'orthographe et de syntaxe sont là aussi fréquentes. Cela nous conduit une fois de plus à privilégier des méthodes d'analyse robustes en contexte difficile. La correction orthographique est assurée dans la limite du possible par une société tiers.

– Des termes différents peuvent être employés pour désigner un même ensemble de produits (par exemple, « canapé », « divan », « sofa » désignent les mêmes produits pertinents). Les internautes peuvent donc employer des termes différents de ceux employés dans les catalogues de produits pour faire référence aux

critères de sélection. Nous proposons une méthode de mise en relation des informations des catalogues de produits avec un lexique de termes généralistes.

– Les critères peuvent porter sur les couleurs désirées. Dans ce cadre, les personnes en charge du marketing se révèlent souvent dotées d'une grande imagination. Il n'est ainsi pas rare de pouvoir acheter des canapés « chocolats » ou bien « lave ». Si ces couleurs présentent l'avantage d'être exotiques, les reconnaître pose une difficulté supplémentaire lorsque l'internaute recherche un canapé « marron » ou « rouge ». Au delà de la recherche de produits, nous abordons cette problématique dans une vision plus globale de recherche de couleurs disponibles dans une base de données.

– Les critères peuvent aussi être complexes : expressions de limites (par exemple, « un canapé à moins 3000 euros », « une voiture entre 20 000 et 30 000 euros »), expressions d'approximation (par exemple, « une valise à 100 euros environ »). Une part d'information implicite peut également être véhiculée par l'expression des critères. Si l'internaute recherche « une valise à 100 euros », une valise à 99 euros peut le satisfaire. Il s'agit d'un cas d'approximation implicite.

Chapitre 1

Préliminaires

Dans ce chapitre, nous définissons quelques notions relatives à la linguistique, la lexicologie, la sémantique, aux suites et aux ensembles, que nous utilisons dans les chapitres suivants.

1.1 Notions linguistiques

Nous commençons par définir les notions clés de langues et langages pour la linguistique.

Définition 1.1 (*langage*)

Un langage désigne un code permettant la communication.

Définition 1.2 (*langage naturel, langue*)

Un langage naturel (ou langue) désigne un code parlé, mimé ou écrit permettant la communication entre individus.

Bien que notre travail porte sur le traitement automatique de langues, l'étude formelle de la syntaxe des langages peut aider à proposer des traitements pour la syntaxe des langues. Les notions suivantes sont utiles à cette fin.

Définition 1.3 (*symbole*)

Un symbole en linguistique correspond à la plus petite unité constituant le langage.

Définition 1.4 (*caractère*)

Un caractère correspond à une typographie d'un symbole.

Définition 1.5 (*alphabet*)

Un alphabet est un ensemble de symboles utilisés dans une langue.

Définition 1.6 (*lettre*)

Une lettre est un symbole appartenant à l'alphabet. On parle aussi de symbole alphabétique.

Exemple 1.1

La langue française est constituée de 26 symboles alphabétiques réunis dans un alphabet allant des symboles a à z. Chacun d'eux constitue une lettre de l'alphabet de la langue française. Chacune de ces lettres peut être représentée typographiquement de plusieurs manières. En informatique, on distingue deux typographies par la casse de la lettre. Cette distinction donne pour la lettre a, les deux caractères 'a' et 'A'. A ces deux représentations typographiques de base s'ajoutent celles comportant des diacritiques, ce qui donne les caractères 'à' et 'â' dans notre exemple.

Nous étudions dans nos travaux les liens existants entre la langue écrite et la langue parlée dans le but de proposer des solutions au traitements de fautes d'orthographe. La notion de phonème est centrale.

Définition 1.7 (*phonème*)

Un phonème est une unité de son d'une langue, d'un langage naturel.

A un autre niveau de l'étude linguistique, l'étude de la morphologie est utile afin d'identifier les familles de mots partageant des sens similaires mais différents en raison des règles syntaxiques et grammaticales.

Définition 1.8 (*morphème*)

Le morphème désigne une unité porteuse de sens dans une langue, un langage naturel. Il est constitué d'une suite finie de symboles appartenant à l'alphabet de

la langue.

Définition 1.9 (*morphème libre, morphème lié*)

Un morphème est dit libre s'il est autonome, c'est-à-dire s'il constitue un mot de la langue. Un morphème est dit lié sinon. Un morphème libre se compose d'un ou plusieurs morphèmes liés.

Définition 1.10 (*morphème radical, radical*)

Le radical est le morphème lié d'un morphème libre se trouvant dans l'ensemble des morphèmes libres d'une famille de morphèmes libres.

Définition 1.11 (*flexion*)

Une flexion (ou morphème flexionnel) est un morphème lié correspondant à une modification subie par le radical selon le genre, nombre et cas du morphème libre ainsi créé.

Définition 1.12 (*dérivation*)

Une dérivation (ou morphème dérivationnel) est un morphème lié correspondant à une modification subie par le radical afin de former un nouveau morphème libre.

Définition 1.13 (*racine, racinisation*)

La racine est le résultat d'une racinisation. Cette opération consiste à identifier parmi les morphèmes liés constituant chaque morphème libre, ceux supprimables tout en garantissant la conservation du sens de départ. Cela consiste pour les langues que nous étudions à supprimer les morphèmes flexionnels et dérivationnels suffixés, porteurs d'informations de nature grammaticale et à conserver le radical et les morphèmes dérivationnels affixés porteur d'informations de nature sémantique.

Exemple 1.2

Morphèmes liés				*Morphème libre*	*Racine*
Radical	*Morphème dérivationnel*		*Morphème flexionnel*	-	-
-	*préfixe*	*suffixe*	-	-	-
jou			*ons*	*jouons*	*jou*
jou			*er*	*jouer*	*jou*
fai	*dé*		*re*	*défaire*	*défai*
fai			*t*	*fait*	*fai*
agré		*able / ment*		*agréablement*	*agré*

Les différents cas étudiés sont donnés sur chaque ligne. Chaque morphème libre donné dans l'avant dernière colonne est issu de l'association du radical donné dans la première colonne avec les morphèmes liés donnés dans les colonnes intermédiaires. La dernière colonne donne la racine identifiée par racinisation.

1.2 Notions lexicologiques

Il est important de pouvoir organiser, regrouper, indexer les morphèmes libres constituant une langue selon leur appartenance à une même famille de mots. Les notions suivantes nous permettent de formaliser cela.

Définition 1.14 (*lexique*)

Un lexique est un inventaire, une liste de morphèmes libres.

Définition 1.15 (*lexème, unité lexicale*)

Un lexème (ou unité lexicale) désigne un morphème libre appartenant à un lexique.

Définition 1.16 (*lemme*)

Le lemme est le représentant canonique de lexèmes résultants de flexions d'un même morphème libre.

1.3 Notions sémantiques

Les sens véhiculés par les mots tiennent une place importante dans notre analyse. La notion de signe linguistique permet d'établir formellement le lien entre un mot et son/ses sens.

Définition 1.17 (*signe linguistique*)

Un signe linguistique désigne une unité autonome d'expression d'un langage.

Définition 1.18 (*signifié*)

Le signifié désigne le concept associé à un signe linguistique.

Définition 1.19 (*signifiant*)

Le signifiant désigne la forme et l'aspect matériel d'un signe linguistique.

Les concepts de signe linguistique, signifiant et signifié sont des fondements de la linguistique moderne (DE SAUSSURE 1916). Le signe linguistique est vu comme la liaison d'un signifiant et d'un signifié. Les liaisons existantes entre signifiants et signifiés sont différentes selon la langue et le moyen de communication utilisés (langue parlée ou écrite).

Dans le cadre d'une langue écrite, le signifiant d'un signe linguistique est un morphème libre. Dans le cadre d'une langue parlée, le signifiant est une succession de sons.

1.4 Convention d'écriture de suites

Nous avons besoin de définir des suites finies. Nous adoptons une convention d'écriture dans laquelle les suites sont définies explicitement et identifiées par une écriture en fonte grasse. Par convention également, les éléments des suites sont dénotés par la même écriture que la suite en minuscule avec un indice.

Exemple 1.3

Nous définissons une suite finie

$$\mathbf{M} = (m_i)_{[\![1,|\mathbf{M}|]\!]},$$

1.5 Convention d'écriture d'ensembles

Nous avons besoin de définir des ensembles associés ou non à des suites finies. Nous adoptons une convention d'écriture dans laquelle les ensembles sont définis par une écriture en fonte non grasse. Lorsque l'ensemble est défini implicitement au travers d'une suite, nous conservons la même notation que pour la suite mais en fonte non grasse. Par convention également, les éléments des ensembles sont dénotés par la même écriture que l'ensemble en minuscule.

Exemple 1.4

Nous définissons implicitement, à partir de la suite $\mathbf{M}$*, un ensemble* M *tel que*

$$M = \{m \mid \exists i \in [\![1, |\mathbf{M}|]\!], m_i = m\}$$

Il est de ce fait possible de définir la suite $\mathbf{M}$ *en tant qu'application dans* M *:*

$$\mathbf{M} : [\![1, |\mathbf{M}|]\!] \to M$$

1.6 Notions ensemblistes

Les notions de partie, ensemble de parties, ensemble des parties et ensembles disjoints sont des notions ensemblistes de base.

Définition 1.20 (*Partie*)

Une partie d'un ensemble X *est un ensemble* P *tel que* $P \subseteq X$*. On parle également de sous-ensemble.*

Définition 1.21 (*Ensemble de parties d'un ensemble*)

Un ensemble de parties d'un ensemble X *est un ensemble* ρ *dont les éléments sont des parties de l'ensemble* X*. Cet ensemble possède la propriété que pour tout ensemble* X*, on a* $|\rho| \leq 2^{|X|}$*.*

Définition 1.22 (*Ensemble des parties*)

L'ensemble des parties d'un ensemble X *est l'ensemble* $P(X)$ *de parties de* X *tel que* $P(X) = \{P \mid P \subseteq X\}$*. Pour tout ensemble* X*, on a* $|P(X)| = 2^{|X|}$*. L'ensemble des*

parties de X peut ainsi être noté 2^X. En d'autres termes, il s'agit de l'ensemble de parties de X de cardinal maximal.

Définition 1.23 (*Ensembles disjoints*)

Deux ensembles X et Y sont disjoints si et seulement si aucun des deux ensembles ne possède un élément en commun avec l'autre : $X \cap Y = \emptyset$

1.7 Relations ensemblistes

Nous décrivons à présent des notions de bases en terme de relations entre éléments d'ensembles. Les relations de type binaire sont particulièrement utiles dans notre travail.

Définition 1.24 (*Relation binaire*)

Une relation binaire R d'un ensemble X dans un ensemble Y est une opération mathématique associant des éléments de X à des éléments de Y. L'association d'un élément $x \in X$ à un élément $y \in Y$ est noté xRy. Formellement, nous définissons un ensemble $G_r \subseteq X \times Y$, dénoté graphe de la relation tel que :
$(x, y) \in G_r \Leftrightarrow xRy$
On dit que la relation R est vérifiée pour les couples $(x, y) \in G_r$.

Définition 1.25 (*Relation surjective*)

Une relation binaire R d'un ensemble X dans un ensemble Y est surjective si et seulement si : $\forall y \in Y, \exists x \in X, xRy$

Définition 1.26 (*Relation injective*)

Une relation binaire R d'un ensemble X dans un ensemble Y est injective si et seulement si : $\forall (x_1, x_2) \in X^2, \forall y \in Y, \big((x_1 R y \wedge x_2 R y) \Rightarrow (x_1 = x_2)\big)$

Définition 1.27 (*Relation bijective*)

Une relation binaire R d'un ensemble X dans un ensemble Y est bijective si et seulement si elle est injective et surjective, soit : $\forall y \in Y, \exists! x \in X, xRy$

Définition 1.28 (*Application*)

Une application est une relation binaire R d'un ensemble X dans un ensemble Y telle que : $\forall x \in X, \exists! y \in Y, xRy$

Un autre type de notation peut également être utilisé pour ce type de relation :

$$\begin{array}{rcl} R & : & X \to Y \\ & & x \mapsto y \end{array}$$

Définition 1.29 (*Application bijective*)

Une application bijective est une relation binaire R qui possède à la fois la propriété de bijection et celle d'être une application.

Définition 1.30 (*Relation de pré-ordre*)

Une relation de pré-ordre R d'un ensemble X dans lui-même est une relation de l'ensemble X dans lui-même vérifiant :
réflexivité : $\forall x \in X, xRx$
transitivité : $\forall (x,x',x'') \in X^3, \big((xRx' \wedge x'Rx'') \Rightarrow xRx''\big)$

Définition 1.31 (*Relation d'ordre*)

Une relation d'ordre R d'un ensemble X dans lui-même est une relation de l'ensemble X dans lui-même vérifiant :
réflexivité : $\forall x \in X, xRx$
antisymétrie : $\forall (x,x') \in X^2, \big((xRx' \wedge x'Rx) \Rightarrow (x = x')\big)$
transitivité : $\forall (x,x',x'') \in X^3, \big((xRx' \wedge x'Rx'') \Rightarrow xRx''\big)$

Définition 1.32 (*Relation d'ordre strict*)

Une relation d'ordre strict R d'un ensemble X dans lui-même est une relation de l'ensemble X dans lui-même vérifiant :
irréflexivité : $\forall x \in X, \neg(xRx)$
asymétrie : $\forall (x,x') \in X^2, \big(xRx' \Rightarrow \neg(x'Rx)\big)$
transitivité : $\forall (x,x',x'') \in X^3, \big((xRx' \wedge x'Rx'') \Rightarrow xRx''\big)$

La figure 1.1 donne les relations de dépendances entre les relations ensemblistes définies dans cette section. La notion de relation binaire est une notion centrale ici.

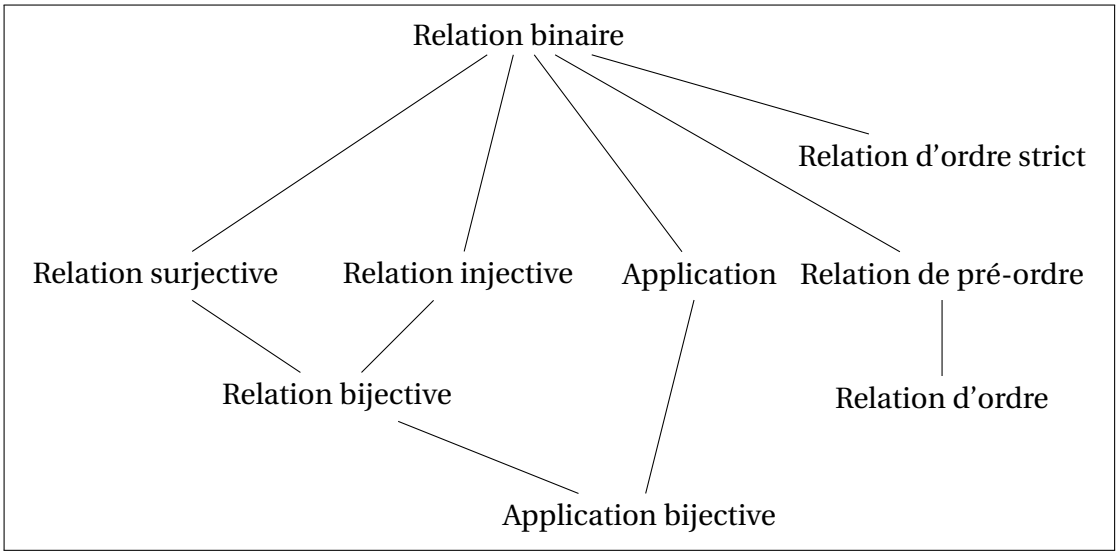


FIGURE 1.1 – Relations de dépendance entre les relations ensemblistes. Une notion dans le graphe dépend de l'ensemble des notions placées plus haut dans le graphe auxquelles elle est liée par un trait.

1.8 Conclusion

Les différentes notions présentées dans ce chapitre nous permettent de formaliser les informations nécessaires pour construire des représentations de textes et de mots et définir des similarités entre elles.

Cela nous est particulièrement utile dans la première partie de cette thèse.

Première partie

Représentations et similarités : des textes aux mots

Chapitre 2

Relations de dissimilarité et similarité

Nous établissons, dans nos travaux, des relations de dissimilarités et similarités entre des textes, des mots et même des couleurs. Ces notions sont centrales dans cette thèse et nous consacrons ce chapitre à leur présentation.

2.1 Définitions

Nous commençons par poser les définitions de ces notions.

Définition 2.1 (*Dissimilarité*)

Une dissimilarité est une application d'un ensemble E dans $\mathbb{R}^+$ telle que

$$diss : E \times E \rightarrow \mathbb{R}^+$$

vérifiant les deux propriétés :

symétrie : $\forall (e, e') \in E^2, diss(e, e') = diss(e', e)$
minimalité : $\forall e \in E, diss(e, e) = 0$

Définition 2.2 (*Indice de distance*)

Un indice de distance, que nous notons $idist$, est une dissimilarité étendant la propriété de minimalité à celle de séparation :

$\forall (e,e') \in E^2, idist(e,e') = 0 \Leftrightarrow e = e'$

Définition 2.3 (*Distance*)

Une distance, que nous notons $dist$, est un indice de distance vérifiant la propriété d'inégalité triangulaire :
$\forall (e,e',e'') \in E^3, dist(e,e') \leq dist(e,e'') + dist(e'',e')$

Définition 2.4 (*Espace métrique*)

Un espace métrique (E,d) est un ensemble E muni d'une distance d.

Définition 2.5 (*Similarité*)

Une similarité est une application d'un ensemble E dans un intervalle $[0, Smax]$ *telle que*

$$sim : E \times E \rightarrow [0, Smax]$$

vérifiant les deux propriétés :

symétrie : $\forall (e,e') \in E^2, sim(e,e') = sim(e',e)$
maximum : $\forall e \in E, sim(e,e) = Smax$

S'il existe un maximum, noté Dmax, pour la dissimilarité utilisée, il est possible de définir une application Φ (resp. Φ') de $[0, Dmax]$ dans $[0, Smax]$ (resp. de $[0, Smax]$ dans $[0, Dmax]$) telle que :

$$\begin{aligned} \Phi &: [0, Dmax] \rightarrow [0, Smax] \\ &\quad diss(e,e') \mapsto sim(e,e') \\ \Phi' &: [0, Smax] \rightarrow [0, Dmax] \\ &\quad sim(e,e') \mapsto diss(e,e') \end{aligned}$$

et telle que

$$\Phi(\Phi'(x)) = x$$
$$\Phi'(\Phi(x)) = x$$

Des définitions de Φ et Φ' couramment utilisées pour leur simplicité sont les suivantes :

$$\begin{aligned}\Phi(x) &= Smax - Smax\frac{x}{\text{Dmax}}\\ \Phi'(x) &= Dmax - \frac{xDmax}{Smax}\end{aligned}$$

La valeur usuellement choisie pour $Smax$ est 1.

2.2 Distances et similarités vectorielles

Définition 2.6 (*Espace vectoriel réel*)

Un espace vectoriel réel E est un ensemble d'éléments, notés $\vec{e}$, muni d'une loi associative V des éléments de E dans $\mathbb{R}^{\dim(E)}$:

$$\begin{aligned}V: \quad E &\to \mathbb{R}^{\dim(E)}\\ \vec{e} &\mapsto (e_i)_{[\![1,\dim(E)]\!]}\end{aligned}$$

Remarque 1. Par la suite et sauf mention contraire, nous parlerons d'espace vectoriel pour désigner un espace vectoriel réel.

Définition 2.7 (*Norme d'un espace vectoriel*)

La norme d'un espace vectoriel E, notée $||.||$, est une distance définie sur E telle que :

$$\begin{aligned}||.||: \quad E\times E &\to \mathbb{R}^{+}\\ (A,B) &\mapsto ||\vec{AB}||\end{aligned}$$

Les espaces vectoriels normés, i.e. munis d'une norme, sont des espaces métriques.

En effet, soit un espace vectoriel E normé. Notons $||.||$, la norme dont est muni E. Nous pouvons alors définir une application de $E \times E$ dans $\mathbb{R}^+$ telle que

$$\forall (\vec{e}, \vec{e'}) \in E^2, dist(\vec{e}, \vec{e'}) = ||\vec{e} - \vec{e'}||$$

vérifiant les propriétés de séparation et d'inégalité triangulaire d'après la définition de la norme. Nous prouvons de plus la symétrie :

$$\forall (\vec{e}, \vec{e'}) \in E^2, dist(\vec{e'}, \vec{e}) = ||\vec{e'} - \vec{e}|| = ||-(\vec{e} - \vec{e'})|| = ||\vec{e} - \vec{e'}|| = dist(\vec{e}, \vec{e'})$$

L'application ainsi définie vérifie les trois propriétés ; de ce fait, c'est une distance et l'espace vectoriel normé est métrique.

Remarque 2. Par la suite et sauf mention contraire, nous emploierons le terme d'espace vectoriel pour désigner un espace vectoriel normé.

De nombreuses relations de dissimilarités ou de similarités peuvent être appliquées sur des espaces vectoriels qu'ils soient de dimensions finies ou infinies. Certaines sont couramment employées dans l'état de l'art. Nous consacrons cette section de ce chapitre à leur présentation. Nous commençons par la distance la plus simple pour finir par la plus complexe.

2.2.1 Distance de Manhattan (ou 1-distance)

Il s'agit d'une distance simple à calculer sur un espace vectoriel de dimension finie. Elle se base sur la valeur absolue de la différence de chaque coordonnée des vecteurs considérés.

Définition 2.8 (*distance de Manhattan*)

Soient $\vec{e}$ et $\vec{e'}$, deux éléments de l'espace vectoriel E tels que

$$\vec{e} = (e_i)_{[\![1,\dim(E)]\!]} \text{ et } \vec{e'} = (e'_i)_{[\![1,\dim(E)]\!]}$$

La distance de Manhattan entre $\vec{e}$ et $\vec{e'}$ s'exprime alors par :

$$\text{Manhattan}(\vec{e}, \vec{e'}) = \sum_{i=1}^{\dim(E)} |e_i - e'_i|$$

2.2.2 Distance euclidienne (ou 2-distance)

La distance euclidienne entre deux points du plan est la longueur du segment qui les relie.

Définition 2.9 (*distance euclidienne*)

La distance euclidienne entre $\vec{e}$ et $\vec{e'}$ s'exprime par :

$$\text{Euclidienne}(\vec{e}, \vec{e'}) = \sqrt{\sum_{i=1}^{\dim(E)} (e_i - e'_i)^2}$$

2.2.3 Distance de Minkowski (ou n-distance)

Les deux distances vues précédemment peuvent être généralisées en une distance unique appelée distance de Minkowski. Celle-ci accepte en paramètre l'ordre de la norme utilisée noté n. Lorsque l'ordre est de 1, la distance de Minkowski est équivalente à la distance de Manhattan et lorsqu'il est de 2, elle est équivalente à la distance euclidienne.

Définition 2.10 (*distance de Minkowski*)

La distance de Minkowski entre $\vec{e}$ et $\vec{e'}$ s'exprime par :

$$\text{Minkowski}(\vec{e}, \vec{e'}, n) = \sqrt[n]{\sum_{i=1}^{\dim(E)} |e_i - e'_i|^n}$$

2.2.4 Distance de Tchebychev

Lorsque l'ordre de la norme tend vers l'infini, la distance de Minkowski devient équivalente à la distance de Tchebychev.

Définition 2.11 (*distance de Tchebychev*)

La distance de Tchebychev entre $\vec{e}$ et $\vec{e'}$ s'exprime par :

$$\text{Tchebychev}(\vec{e},\vec{e'}) = \lim_{n\to\infty} \text{Minkowski}(\vec{e},\vec{e'},n) = \max_{i=1}^{\dim(E)} |e_i - e'_i|$$

2.2.5 Similarité du cosinus

Les distances vues précédemment ont une grande sensibilité aux normes des vecteurs utilisés puisqu'elles se basent directement sur celles-ci pour évaluer une distance. Il peut être souhaitable, et c'est souvent le cas pour des distances de textes, de calculer une distance qui ne soit pas sensible à la norme mais plutôt à la direction des vecteurs. La distance ou similarité du cosinus est dans ce cas un choix pertinent.

Définition 2.12 (*similarité du cosinus*)

La similarité du cosinus donne une mesure de similarité entre deux vecteurs basée sur le cosinus de l'angle formé par ceux-ci.

$$\text{Cos}(\vec{e},\vec{e'}) = \frac{\vec{e}.\vec{e'}}{||\vec{e}||.||\vec{e'}||} = \frac{\sum_{i=1}^{\dim(E)} e_i.e'_i}{\sqrt{\sum_{i=1}^{\dim(E)} (e_i)^2}\sqrt{\sum_{i=1}^{\dim(E)} (e'_i)^2}}$$

Dans un espace vectoriel quelconque, la similarité du cosinus est comprise dans l'intervalle $[-1,1]$ avec pour valeurs remarquables -1 si et seulement si les directions des vecteurs s'opposent, 0 si les vecteurs sont orthogonaux et 1 s'ils ont la même direction. Dans un espace vectoriel E tel que $\forall \vec{e} \in E, e_i \geq 0$, cet intervalle se réduit à celui des mesures de similarité à savoir $[0,1]$.

La similarité du cosinus est une mesure liée à la distance euclidienne lorsque les vecteurs sont normalisés. En effet, soient $\vec{e}$ et $\vec{e'}$ deux vecteurs normalisés. La norma-

lisation des vecteurs implique que :

$$||\vec{e}|| = ||\vec{e'}|| = 1$$
$$\text{Euclidienne}(\vec{e},\vec{e'})^2 = ||(\vec{e}-\vec{e'})||^2$$
$$\text{Euclidienne}(\vec{e},\vec{e'})^2 = ||\vec{e}||^2 + ||\vec{e'}||^2 - 2.||\vec{e}||.||\vec{e'}||.\text{Cos}(\vec{e},\vec{e'})$$
$$\text{Euclidienne}(\vec{e},\vec{e'})^2 = 2 - 2.\text{Cos}(\vec{e},\vec{e'})$$

Ainsi, lorsque les vecteurs sont normalisés, la distance euclidienne et la similarité du cosinus vont produire des résultats similaires en terme de discrimination des points de l'espace vectoriel. Les algorithmes d'apprentissage qui produisent un bon résultat avec la similarité du cosinus produisent donc également un bon résultat avec une distance euclidienne normalisée et inversement.

Chapitre 3

Représentation des textes

Dans ce chapitre, nous décrivons les diverses techniques de représentations de textes. Une place importante est accordée à la famille des représentations vectorielles.

3.1 Définitions et principes de la représentation des textes

Commençons par définir la notion de représentation informatique d'un texte.

Définition 3.1 (*représentation informatique d'un texte*)

La représentation informatique d'un texte (ou par métonymie, représentation d'un texte) est une description, un modèle, une image d'un texte traitable par des processus informatiques. Cette image peut être d'une fidélité variable au texte d'origine et plus ou moins riche en informations selon les traitements informatiques que l'on souhaite réaliser sur le texte par le biais de son image.

Les types de représentations sont nombreux et offrent des formes différentes. Chacune possède un degré différent de complexité, des représentations les plus triviales cherchant uniquement à modéliser la typographie du texte aux représentations les plus complexes visant à modéliser sa sémantique.

Nous présentons dans ce chapitre quelques représentations de textes parmi les plus largement utilisées et indiquons les liens existant entre elles. Nous montrons

comment obtenir chacune d'elles à partir d'une représentation plus élémentaire. Cela nous conduit à présenter en premier lieu les représentations les plus aisées à construire pour aller vers les plus difficiles à obtenir.

Remarque 3. Pour les exemples suivants de représentations d'un texte, nous utilisons le texte T_1 : « Je suis très satisfait de vos services. Continuez comme cela, c'est très bien ! ! ! ! », comme texte à représenter.

3.2 Les représentations typographiques de textes

Nous commençons dans cette partie par les représentations typographiques des textes. Ce sont les plus simples à construire et les plus largement utilisées pour représenter un texte.

3.2.1 Représentation d'un texte par suite finie de caractères

L'élément basique de la typographie étant le caractère, nous commençons par la représentation se basant sur celui-ci.

Définition 3.2 (*représentation d'un texte par suite finie de caractères*)

La représentation d'un texte par suite finie de caractères représente un texte constitué de caractères c_i comme une suite finie de caractères **C**.

$$\mathbf{C} = (c_i)_{[\![1,|\mathbf{C}|]\!]}$$

Exemple 3.1

La représentation par suite finie de caractères de T_1 est de la forme
$\mathbf{C}_1$ = *('j', 'e', ' ','s', 'u', 'i', 's', ' ', 't', 'r', 'è', 's', ' ', 's', 'a', 't', 'i', 's', 'f', 'a', 'i', 't', ' ', 'd', 'e', ' ', 'v', 'o', 's', ' ', 's', 'e', 'r', 'v', 'i', 'c', 'e', 's', '.', ' ', 'c', 'o', 'n', 't', 'i', 'n', 'u', 'e', 'z', ' ', 'c', 'o', 'm', 'm', 'e', ' ', 'c', 'e', 'l', 'a', '\',', ' ', 'c', '\"', 'e', 's', 't', ' ', 't', 'r', 'è', 's', ' ', 'b', 'i', 'e', 'n', ' !', ' !', ' !', ' !')*

Cette représentation est utilisée par défaut par l'ensemble des langages de programmation. Elle doit sa large utilisation à sa simplicité de construction et à sa grande fidélité de représentation typographique du texte. Cependant, l'élément central est

le caractère, élément qui ne porte aucune information directe sur la sémantique du texte. Cette représentation n'est de ce fait pas adaptée pour représenter celle-ci.

Nous distinguons deux types de caractères constituant les textes dans la représentation d'un texte par suite de caractères :

1. les caractères alphabétiques qui correspondent aux symboles alphabétiques de la langue.
2. les caractères non alphabétiques qui correspondent à des ponctuations ou plus généralement à des marques de structuration du langage.

Si toutes les langues présentent des caractères alphabétiques utilisés de manière significative dans les textes, il n'en est pas de même pour les caractères non alphabétiques.

3.2.2 Représentation d'un texte par suite finie de mots

Le mot constitue un élément typographique plus complexe, présentant sa propre sémantique dès lors qu'on le considère comme un signe linguistique (DE SAUSSURE 1916).

Pour la plupart des langues occidentales et particulièrement en français, la représentativité des caractères non alphabétiques est suffisante pour les utiliser comme séparateurs dans la construction de la représentation par suite finie de mots.

Définition 3.3 (*représentation d'un texte par suite finie de mots*)

La représentation d'un texte par suite finie de mots modélise un texte par une suite finie **M** *de mots* m_i *à partir de la suite finie de caractères* **C**.
Soit $Alpha$, *l'ensemble des caractères définissant l'alphabet de la langue. Soient* debutMot *et* finMot, *les fonctions donnant les indices des caractères de la suite* **C** *correspondant au début et à la fin du mot dont l'indice est spécifié en entrée,*

nous les définissons formellement :

$$\begin{aligned}
\text{debutMot} &: [\![1,|\mathbf{M}|]\!] \to [\![1,|\mathbf{C}|]\!] \\
\text{debutMot}(1) &= \min\{i \mid c_i \in Alpha\} \\
\text{debutMot}(n+1) &= \min\{i \mid i > \text{finMot}(n) \wedge c_i \in Alpha\} \\
\text{finMot} &: [\![1,|\mathbf{M}|]\!] \to [\![1,|\mathbf{C}|]\!] \\
&\quad n \mapsto \min\{i \mid i > \text{debutMot}(n) \wedge (i+1 > |\mathbf{C}| \vee c_{i+1} \notin Alpha)\}
\end{aligned}$$

Nous pouvons alors définir formellement la suite de mots :

$$\mathbf{M} = (m_i)_{[\![1,|\mathbf{M}|]\!]} \text{ avec } m_i = (c_j)_{[\![\text{debutMot}(i),\text{finMot}(i)]\!]}$$

Exemple 3.2

La représentation par suite finie de mots de T_1 est de la forme
$\mathbf{M}_1$ = ('je', 'suis', 'très', 'satisfait', 'de', 'vos', 'services', 'continuez', 'comme', 'cela', 'c\'est', 'très', 'bien')

Pour les langues n'utilisant pas ou peu de caractères non alphabétiques comme certaines langues orientales, la définition des fonctions $debutMot$ et $finMot$ ne peut plus se faire au travers des caractères non alphabétiques. Notre étude ne porte pas sur ces cas complexes.

La représentation par suite finie de mots véhicule une quantité d'informations sémantiques plus importante que celle par suite finie de caractères. L'utilisation des mots comme éléments centraux permet d'exploiter la sémantique véhiculée par ceux-ci (les signifiés des mots) pour transcrire/représenter la sémantique du texte non pas de manière directe mais indirectement par leurs intermédiaires. Cette représentation ne permet pas la restitution du texte d'origine de par la perte des caractères de ponctuation mais cela ne pose pas nécessairement de problèmes pour le traitement du texte.

L'utilisation de tous les mots présents dans un texte engendre une taille de représentation considérable, pouvant être difficile à traiter dans certains cas. Dans l'exemple minimaliste du texte T_1, la représentation atteint déjà une taille de 13 éléments. Il existe ainsi des représentations basées sur les mots des textes qui, contrairement à la

représentation par suite de mots, cherchent à optimiser le rapport existant entre le nombre de mots à utiliser dans la représentation et l'exactitude de la sémantique du texte à représenter par l'intermédiaire de leurs signifiés. Ce sont là les fondements et paris faits par les représentations de type vectoriel que nous abordons dans la section suivante. Cependant, cette réduction du nombre de mots des représentations vectorielles de textes, couramment désignée par réduction de l'espace vectoriel, peut avoir un impact fort sur la qualité finale des traitements : si les textes ne sont pas correctement représentés, il est peu vraisemblable que les résultats obtenus soient probants car la représentation ne reflétera pas le sens du texte. Certains chercheurs préconisent de ne pas systématiquement employer ces méthodes de réduction mais au contraire de réfléchir à leur pertinence et au fait que des mots supprimés traditionnellement des représentations peuvent être primordiaux pour des traitements proches de la sémantique (RILOFF 1995).

3.3 Les représentations vectorielles de textes

Contrairement aux représentations par suites que nous avons présentées précédemment, les représentations vectorielles de textes offrent naturellement des relations de dissimilarités ou similarités entre les textes représentés. Les relations usuelles des espaces vectoriels présentées au chapitre 2 peuvent être employées. Cela constitue un avantage de ce type de représentations pour nos traitements surtout lorsque les représentations utilisées tiennent compte de l'éloignement sémantique et linguistique existant entre les textes.

3.3.1 Représentation vectorielle standard d'un texte

Commençons d'abord par définir la représentation vectorielle la plus simple et la plus naturelle de la famille des représentations vectorielles.

Définition 3.4 (*représentation vectorielle standard d'un texte*)

*La représentation vectorielle standard d'un texte (*SALTON, WONG *et* YANG *1975) représente un texte constitué d'un ensemble M de mots m distincts par un vecteur tel que la valeur de chaque composante représentative d'un mot m distinct*

du texte vaut le nombre d'occurrences du mot m dans la suite de mots ***M****.*
Le nombre d'occurrences d'un mot donné dans le texte est dénoté

$$|m|_{VS} = |\{i \in [\![1, |\mathbf{M}|]\!] \mid \ m_i = m\}|$$

La représentation vectorielle standard est alors donnée par le graphe de la fonction $|.|_{VS}$.

Exemple 3.3

Le graphe G_1^{VS} *de la fonction* $|.|_{VS}$ *donnant la représentation vectorielle standard de* T_1 *est de la forme :*
G_1^{VS} *= { ('je', 1), ('suis', 1), ('très', 2), ('satisfait', 1), ('de', 1), ('vos', 1), ('services', 1), ('continuez', 1), ('comme', 1), ('cela', 1), ('c\'est', 1), ('bien', 1) }*

Dans la représentation vectorielle standard, la sémantique du texte représentée par les signifiés des mots de ce dernier est indépendante de la place occupée par les mots mais uniquement dépendante de la fréquence d'apparition du signifié. Dans l'exemple du texte T_1, le mot 'très' apparaît deux fois donnant une importance double à son signifié pour le texte par rapport au signifié des autres mots. Même si la taille de la représentation est quelque peu réduite par rapport à la suite de mots (12 éléments au lieu de 13), le rapport évoqué dans la section précédente est encore loin d'être optimal surtout si l'on considère la perte d'informations sur la sémantique du texte liée à l'ordre des mots.

3.3.2 Pondération de l'espace vectoriel

Il peut apparaître assez intuitif que certains mots (et donc leurs signifiés respectifs) d'un texte jouent un rôle prédominant par rapport à d'autres dans l'expression de la sémantique du texte. Cette vision s'oppose à celle d'équi-importance de la représentation vectorielle standard qui considère que tous les mots apportent la même quantité d'informations. Cela conduit dès lors à considérer que certains mots (et à travers eux leurs signifiés) peuvent être éliminés de la représentation du texte sans que la sémantique de celui-ci n'en soit affectée. Dans l'exemple du texte T_1, les signifiés des mots 'satisfait', 'services', 'continuer' et 'bien' sont prédominants tandis que

ceux des mots 'je', 'de', 'comme' sont négligeables. De manière formelle, on considère une application de pondération P :

$$P : M \rightarrow [0,1]$$

$P(m) = 0$ si et seulement si le signifié de m n'a pas d'importance pour le texte représenté par M et $P(m) = 1$ si et seulement si le signifié de m a une importance maximale pour le texte représenté par M.

Nous pouvons dès lors exprimer en fonction de P la représentation vectorielle pondérée tenant compte du degré d'informations sémantiques apportées au texte par chaque mot.

Définition 3.5 (*représentation vectorielle pondérée d'un texte*)

La représentation vectorielle pondérée d'un texte représente un texte constitué d'un ensemble M de mots m distincts par un vecteur tel que la valeur de chaque composante représentative d'un mot m distinct du texte vaut le nombre d'occurrences du mot m dans le texte multiplié par la pondération fournie par l'application P pour le mot m.

$$|m|_{VP} = |m|_{VS} \times P(m)$$

La représentation vectorielle pondérée est alors donnée par le graphe de la fonction $|.|_{VP}$.

Exemple 3.4

Le graphe G_1^{VP} de la fonction $|.|_{VP}$ donnant la représentation vectorielle pondérée de T_1 est de la forme :
$G_1^{VP} = \{('\text{je}', P('\text{je}'))$, $('\text{suis}', P('\text{suis}'))$, $('\text{très}', 2 \times P('\text{très}'))$, $('\text{satisfait}', P('\text{satisfait}'))$, $('\text{de}', P('\text{de}'))$, $('\text{vos}', P('\text{vos}'))$, $('\text{services}', P('\text{services}'))$, $('\text{continuez}', P('\text{continuez}'))$, $('\text{comme}', P('\text{comme}'))$, $('\text{cela}', P('\text{cela}'))$, $('\text{c\'est}', P('\text{c\'est}'))$, $('\text{bien}', P('\text{bien}'))\}$

L'application P la plus simple que l'on puisse envisager est une application à valeur binaire, ne retournant que 1 ou 0. Dans ce cadre, soit le mot est jugé pertinent, soit il est jugé comme ne l'étant pas sans aucune distinction intermédiaire possible. Une approche implémentatoire usuelle consiste à établir une liste de mots à éliminer

(à pondérer de manière nulle), considérée comme peu évolutive et de taille raisonnable (SALTON, WONG et YANG 1975) (SALEM 1987). Le contenu et la longueur de cette liste noire sont différents en fonction de la nature du traitement. Ces mots peu pertinents sont couramment désignés en tant que mots creux. Une seconde approche implémentatoire consiste à utiliser les catégories grammaticales des mots afin de déterminer des catégories à éliminer. Cette approche repose sur une analyse lexicale préalable du texte que nous étudions par la suite plus en détails.

Il est aussi possible de chercher à mesurer plus finement la quantité d'informations apportées par le signifié du mot au texte. Pour cela, on peut avoir recours à des mesures telles que l'entropie proposée par Shannon dans sa théorie de l'information (SHANNON 1948).

Définition 3.6 (*entropie de Shannon*)

$$H(m) = -p(m) \times log\big(p(m)\big)$$

avec $p(m)$, la fonction donnant la probabilité d'apparition du mot m.

Si le texte donné s'inscrit dans un corpus de texte, nous pouvons définir l'application de pondération non plus uniquement en fonction de ce texte mais également en fonction du corpus qui le contient. Formellement, l'application de pondération peut alors s'exprimer comme :

$$P : M \times CORP \rightarrow [0,1]$$

$P(m, CORP) = 0$ si et seulement si le signifié de m n'a pas d'importance pour le texte représenté par M dans le corpus $CORP$ et $P(m, CORP) = 1$ si et seulement si le signifié de m a une importance maximale pour le texte représenté par M dans le corpus $CORP$.

La représentation vectorielle pondérée résultante se définit comme :

Définition 3.7 (*représentation vectorielle pondérée d'un texte en corpus*)

La représentation vectorielle pondérée d'un texte en corpus représente un texte constitué d'un ensemble M de mots distincts par un vecteur tel que la valeur de

chaque composante représentative d'un mot m distinct du texte vaut le nombre d'occurrences du mot m dans le texte multiplié par la pondération trouvée par l'application P pour le mot m dans le texte du corpus $CORP$.

$$|m|_{VPC} = |m|_{VS} \times P(m, CORP)$$

La représentation vectorielle pondérée d'un texte en corpus est alors donnée par le graphe de la fonction $|.|_{VPC}$.

Dans ce contexte, pondérer un terme en fonction de sa fréquence dans le texte est une solution envisageable. Cette technique fut proposée par Hans Peter Luhn en 1958.

Définition 3.8 (*Term frequency (TF)* (LUHN *1958*))

$$TF(m) = \frac{|m|_{VS}}{|\mathbf{M}|}$$

Cette technique présente cependant l'inconvénient de privilégier fortement les mots les plus occurrents dans un corpus alors qu'il ne sont pas nécessairement les plus informatifs. Ils peuvent en effet correspondre simplement à des termes grammaticaux très communs. Ce problème peut être contourné par l'emploi de l'inverse de la fréquence du terme dans le corpus. Cette technique fut proposée par Karen Sparck Jones en 1972.

Définition 3.9 (*Inverse document frequency (IDF)* (SPARCK JONES *1972*))

$$IDF(m, CORP) = log\frac{|CORP|}{|\{\mathbf{M} \in CORP | m \in \mathbf{M}\}|}$$

L'association des méthodes *term frequency* et *inverse document frequency* est une méthode également proposée par Karen Sparck Jones en 1972 et reprise par la suite par Salton et Buckley dans la recherche d'informations (SALTON et BUCKLEY 1988). Le but est de combiner les capacités discriminantes des deux formules pour obtenir les meilleurs pondérations sur les termes les plus pertinents pour la recherche d'informations. Cela en fait encore aujourd'hui une référence en matière de techniques de pondération pour la recherche d'informations.

Définition 3.10 (*Term frequency, inverse document frequency (TF.IDF)* (SPARCK JONES *1972*))

$$TF.IDF(m, CORP) = TF(m).IDF(m, CORP)$$

3.3.3 Méthodes de réduction de l'espace vectoriel

Si la pondération présentée dans la sous-section précédente contribue à la réduction de l'espace vectoriel par la suppression des mots les moins représentatifs de la sémantique du texte, d'autres méthodes basées sur l'analyse lexicale et morphologique des textes sont également couramment employées dans ce but. Les plus répandues, la lemmatisation et la racinisation sont des méthodes qui cherchent à identifier les mots de la langue différents à cause des règles syntaxiques et grammaticales mais ayant une sémantique identique. Ces mots sont alors représentés par un signifiant unique au niveau de la représentation vectorielle. De manière plus poussée, une exploitation des règles linguistiques de synonymie peut être envisagée dans le but de réduire encore davantage la dimension de l'espace vectoriel de la représentation.

Analyse morphologique : réduction de l'espace par racinisation

L'analyse morphologique a pour but d'analyser les mots d'un texte en tant que morphèmes libres constitués de racines et morphèmes liés. Il s'agit donc du procédé visant à associer chacun des mots d'un texte aux racines et morphèmes liés les constituant.

L'analyse morphologique des mots d'un texte peut contribuer à la réduction de sa représentation par l'emploi d'une racinisation.

Exemple 3.5

Prenons l'exemple suivant.

Morphèmes liés				*Morphème libre*	*Racine*
Radical	*Morphème dérivationnel*		*Morphème flexionnel*	-	-
-	*préfixe*	*suffixe*	-	-	-
jou			*ons*	*jouons*	*jou*
jou			*er*	*jouer*	*jou*
fai	*dé*		*re*	*défaire*	*défai*
fai			*t*	*fait*	*fai*
agré		*able / ment*		*agréablement*	*agré*
all			*er*	*aller*	*all*
va				*va*	*va*

Les différents cas étudiés sont donnés sur chaque ligne. Chaque morphème libre donné dans l'avant-dernière colonne est issu de l'association du radical donné dans la première colonne avec les morphèmes liés donnés dans les colonnes intermédiaires. La dernière colonne donne la racine identifiée par racinisation.

Dans cet exemple, les mots « défaire » et « fait » possèdent deux racines distinctes (« défai » et « fai ») tandis que les mots « jouons » et « jouer » partagent la même racine (« jou »). La racinisation échoue en revanche à reconnaître « aller » et « va » comme étant deux formes d'un même verbe.

Formellement, nous pouvons définir la racinisation des mots d'un texte comme la constitution de la suite **RAC** à partir de la suite de mots **M** d'un texte :

$$\mathbf{RAC} = (rac_i)_{[\![1,|\mathbf{M}|]\!]} \text{ avec } rac_i = racinisation(m_i)$$

Nous pouvons dès lors définir la représentation vectorielle standard des racines des mots d'un texte

Définition 3.11 (*représentation vectorielle standard des racines des mots d'un texte*)

La représentation vectorielle standard des racines des mots d'un texte représente un texte constitué d'un ensemble RAC de racines rac distincts par un vecteur tel que la valeur de chaque composante représentative d'une racine rac distinct du

texte vaut le nombre d'occurrences de la racine rac dans la suite de racines **RAC**. *Le nombre d'occurrences d'une racine donnée dans le texte est dénoté*

$$|rac|_{VSR} = |\{i \in [\![1, |\mathbf{RAC}|]\!] \mid rac_i = rac\}|$$

La représentation vectorielle standard des racines des mots d'un texte est donnée par le graphe de la fonction $|.|_{VSR}$.

La racinisation n'a besoin d'aucun contexte et utilise uniquement la forme des mots et les règles de la langue utilisée pour en déduire une racine. Elle est ainsi capable d'effectuer des rapprochements entre les mots occupant une fonction grammaticale différente s'ils sont de la même famille. En contrepartie, la racinisation est sensible à la variation du radical rencontrée dans les langues les plus flexionnelles.

Exemple 3.6

Dans l'exemple du texte T_1, l'analyse morphologique donne la suite de morphèmes suivants :
Je suis très satisfait de vos services Continuez comme cela c est très bien
avec les racines soulignées. Dans cet exemple, le problème des langues flexionnelles apparaît à travers le verbe être. Celui-ci est présent sous deux formes différentes : « suis » et « est ». Le processus de racinisation ne les rapproche pas puisque leur radical est différent. Il en est de même pour « ce » et « cela » à cause du phénomène d'élision devant être.

Exemple 3.7

Dans l'exemple du texte T_1, le processus de racinisation conduit à une représentation vectorielle standard des racines dont le graphe est le suivant :
$G_1^{VSR} = \{(Je,1),(suis,1),(très,2),(satisfai,1),(de,1),(vo,1),(service,1),$
$(Continu,1),(comme,1),(ce,1),(c,1)(est,1),(bien,1)\}$

Un autre problème récurrent de la racinisation concerne les règles de la langue à appliquer pour identifier à partir d'un mot son morphème radical. Ces règles conduisent parfois à considérer que deux mots issus de deux familles différentes partagent un radical commun. Dans ce cas, utiliser la racine comme élément de représentation vectorielle conduira à confondre de tels mots.

Les algorithmes de racinisation écrits à partir de l'algorithme Snowball mis au point par Porter (PORTER 1980) sont largement utilisés dans la communauté. Ainsi, le projet Apache Lucène [1] l'utilise pour proposer des outils de racinisation pour de nombreuses langues.

Analyse lexicale : réduction de l'espace par lemmatisation

Contrairement à l'analyse morphologique, l'analyse lexicale analyse les mots d'un texte en tant que morphèmes libres appartenant au lexique de la langue à traiter. Il s'agit donc du procédé visant à associer chacun des mots d'un texte à son ou ses lexèmes correspondant dans le lexique de la langue traitée. Formellement, si l'on dénote L, l'ensemble des lexèmes constituant le lexique de la langue et M, l'ensemble des mots d'un texte, nous pouvons définir de manière basique l'analyse lexicale d'un texte par la fonction :

$$AnalyseLex : M \rightarrow 2^L$$

Ce type d'analyse lexicale présente l'avantage d'être assez simple à mettre en œuvre puisque c'est l'égalité stricte entre le mot et un des lexèmes du lexique qui permet le rapprochement. En contrepartie, le cardinal de l'ensemble des lexèmes trouvés en correspondance pour chaque mot est relativement élevé puisqu'aucune désambiguïsation n'est faite sur le mot.

Cette solution n'est de ce fait bien souvent utilisée que lorsqu'aucun contexte n'est exploitable depuis la représentation choisie pour le texte. Dans les autres cas, une désambiguïsation est réalisée sur le sens des mots afin de réduire au maximum le cardinal de l'ensemble obtenu par l'association. Ce procédé est largement connu et désigné couramment en anglais par *Word Sense Disambiguisation (WSD)*. Formellement, si l'on dénote WSD la méthode de désambiguïsation, nous pouvons alors définir l'analyse lexicale avec désambiguïsation d'un texte par la fonction :

$$AnalyseLexWsd : WSD(M) \rightarrow 2^L$$

1. http ://lucene.apache.org/

L'ensemble des parties résultant de la fonction $AnalyseLexWsd$ est égal ou strictement inclus dans l'ensemble des parties résultant de la fonction $AnalyseLex$.

Une approche classique du problème de désambiguïsation des mots est l'approche grammaticale. Elle consiste à affecter pour chacun des mots d'une phrase, une catégorie grammaticale possible dans la langue choisie selon le mot rencontré et les catégories grammaticales déterminées pour les mots appartenant à son contexte dans le texte. Ce traitement est réalisé par des outils tel que *TreeTagger* (SCHMID 1994), étiqueteur grammatical multilingue réalisé par le laboratoire de l'Université de Stuttgart. Ces outils procèdent au moyen d'une étape préalable d'entraînement sur un corpus de textes de la langue à traiter. Ces textes ont été annotés au préalable, de manière souvent semi-automatique, avec des informations grammaticales sur les différents mots les composants. Le corpus annoté ainsi obtenu est qualifié d'arboré (*Treebank* en anglais) dans le sens où il présente un formatage des textes selon une structure arborescente. Le rôle du processus d'entraînement est double. Premièrement, il identifie des règles dans les enchaînements grammaticaux de la langue à partir des exemples fournis par le corpus. Deuxièmement, il constitue une ressource donnant les triplets (mot, lexème, catégorie grammaticale) à partir des exemples fournis par le corpus. En phase postérieure d'exploitation, l'algorithme identifie les fonctions grammaticales des mots à partir des règles apprises et peut en déduire le ou les lexèmes correspondants à partir de la ressource construite au préalable. Formellement, si l'on dénote par $Gram$, l'ensemble des fonctions grammaticales des mots de la langue considérée, nous pouvons définir l'analyse lexicale d'un texte par la fonction :

$$AnalyseLexGram : M \times Gram \rightarrow 2^L$$

Quelle que soit l'approche retenue, l'analyse lexicale va permettre de réduire l'espace de la représentation vectorielle au moyen d'un processus de lemmatisation.

Définition 3.12 (*Lemmatisation*)

La lemmatisation est le processus visant à identifier à partir d'un lexème du lexique de la langue traitée, sa forme canonique unique correspondante appelée lemme.

Il existe deux familles de méthodes de lemmatisation, appelées lemmatisation en contexte et hors contexte, réalisées respectivement à la suite d'une analyse lexicale avec et sans contexte.

Formellement, nous pouvons définir la lemmatisation des mots d'un texte comme la constitution de la suite de lemmes **LEM** à partir de la suite de mots **M** d'un texte,

$$\mathbf{LEM} = (lem_i)_{[\![1,|\mathbf{M}|]\!]} \text{ avec } lem_i = lemmatisation(m_i)$$

où lem_i représente l'ensemble des lemmes possibles du mot m_i selon la stratégie de désambiguïsation choisie.

Le cardinal de l'ensemble des lemmes pour un mot donné peut être supérieur à 1. Dans ce cas, la représentation vectorielle doit exprimer la non-levée de l'ambiguïté. Une stratégie possible que nous retenons ici est de représenter chaque lemme possible comme attribut du vecteur. Cette stratégie présente l'avantage de pouvoir faire la liaison facilement entre les contextes non-désambiguïsables et ceux désambiguïsables.

Nous pouvons définir la représentation vectorielle standard des lemmes des mots d'un texte comme suit :

Définition 3.13 (*représentation vectorielle standard des lemmes des mots d'un texte*)

La représentation vectorielle standard des lemmes des mots d'un texte représente un texte constitué d'un ensemble LEM de lemmes lem distincts par un vecteur tel que la valeur de chaque composante représentative d'un lemme lem distinct du texte vaut le nombre d'occurrences du lemme lem dans la suite de lemmes **LEM**.
Le nombre d'occurrences d'un lemme donné dans le texte est dénoté

$$|lem|_{VSL} = |\{i \in [\![1,|\mathbf{LEM}|]\!] \mid lem \in lem_i\}|$$

La représentation vectorielle standard des lemmes des mots d'un texte est donnée par le graphe de la fonction $|.|_{VSL}$.

La lemmatisation est capable de rapprocher des mots dont les flexions modifient profondément l'aspect mais pas ceux occupant une fonction grammaticale diffé-

rente même s'ils sont de la même famille, au contraire de la racinisation vue précédemment.

Exemple 3.8

Dans l'exemple du texte T_1, l'analyse lexicale peut conduire à la lemmatisation suivante :
Je être très satisfait de votre service Continuer comme cela c être très bien

Dans cet exemple, le traitement des langues flexionnelles apparaît au travers du verbe être. Celui-ci apparaît sous deux formes différentes : « suis » et « est ». Le processus de lemmatisation les rapproche par leur lemme commun « être ». Cependant, ce rapprochement ne peut s'effectuer de manière certaine si la désambiguïsation est de nature grammaticale car la forme « suis » reste ambiguë : verbe être ou verbe suivre ? Il faut alors employer des techniques sémantiques de désambiguïsation pour lever cette incertitude.

Exemple 3.9

Dans l'exemple du texte T_1, le processus de lemmatisation, appliqué avec une stratégie de désambiguïsation grammaticale, conduit à une représentation vectorielle standard des lemmes dont le graphe est le suivant :
$G_1^{VSL} = \{(Je,1),(suivre,1),(\hat{e}tre,2),(très,2),(satisfait,1),(de,1),(votre,1),(service,1),$
$(Continuer,1),(comme,1),(cela,1),(c,1),(bien,1)\}$

Appliqué avec une stratégie de désambiguïsation sémantique, la représentation vectorielle standard des lemmes donne le graphe suivant :
$G_1^{VSL} = \{(Je,1),(\hat{e}tre,2),(très,2),(satisfait,1),(de,1),(votre,1),(service,1),$
$(Continuer,1),(comme,1),(cela,1),(c,1),(bien,1)\}$

Notons que l'ambiguïté a pour conséquence d'augmenter la taille des représentations des textes.

Analyse lexicale : réduction de l'espace par synonymie

L'analyse lexicale peut également conduire à effectuer une réduction de l'espace vectorielle en utilisant les relations de synonymie connues entre les différents lexèmes

du lexique. Formellement, la synonymie peut être définie par l'opérateur :

$$\begin{array}{rcll} \sim & : & L \times L & \rightarrow & \{Vrai, Faux\} \\ & & l \not\sim l' & \Leftrightarrow & l \text{ et } l' \text{ ne sont pas synonymes} \\ & & l \sim l' & \Leftrightarrow & l \text{ et } l' \text{ sont synonymes} \end{array}$$

Les principales propriétés de cet opérateur sont les suivantes :

1. Il est réflexif : $\forall l \in L, l \sim l$
2. Il est symétrique : $\forall (l, l') \in L^2, \left(l \sim l' \Leftrightarrow l' \sim l\right)$

A partir de cet opérateur, nous pouvons établir assez facilement un graphe dont les sommets sont les différents lexèmes constituants le lexique et les arêtes sont les associations pour lesquelles la relation de synonymie est vérifiée. Dénotons par S, l'ensemble des cliques maximales que nous pouvons former dans ce graphe. Ces cliques maximales caractérisées par une forte connexité symétrique sont usuellement dénotées par l'abréviation anglaise *synsets*. Formellement, nous avons donc :

$$S = \{s \mid s = \{l \mid \forall l' \in s, l' \sim l\}\}$$

Une réduction de l'espace de représentation peut être réalisée en ne représentant plus chaque lexème correspondant aux mots de la phrase mais plutôt chaque *synset* correspondant aux mots de la phrase.

Il est assez usuel de choisir le lexème le plus petit dans l'ordre alphabétique comme représentant d'un *synset* bien qu'il soit possible d'utiliser n'importe quel système d'identification unique.

Formellement, nous pouvons définir une suite de synsets **SYN** présents dans le texte à partir de la suite de mots **M** d'un texte,

$$\mathbf{SYN} = (syn_i)_{[\![1,|\mathbf{M}|]\!]} \text{ avec } syn_i = s \mid \exists l \in s, l = AnalyseLex(m_i)$$

Nous pouvons définir la représentation vectorielle standard des *synsets* des mots d'un texte comme suit :

Définition 3.14 (*représentation vectorielle standard des synsets des mots d'un texte*)

La représentation vectorielle standard des synsets des mots d'un texte représente un texte constitué d'un ensemble SYN de synsets syn distincts par un vecteur tel que la valeur de chaque composante représentative d'un synset syn distinct du texte vaut le nombre d'occurrences du synset syn dans la suite de synsets **SYN**.
Le nombre d'occurrences d'un synset donné dans le texte est dénoté,

$$|syn|_{VSS} = \left|\left\{i \in [\![1, |\mathbf{SYN}|]\!] \mid syn_i = syn\right\}\right|$$

La représentation vectorielle standard des synsets des mots d'un texte est alors donnée par le graphe de la fonction $|.|_{VSS}$.

Une approche statistique de la représentation vectorielle sémantique peut également être mise en œuvre à partir des mots de la représentation vectorielle standard. La méthode *Latent Semantic Analysis* (LSA) (DEERWESTER et al. 1990) en est un exemple. La méthode LSA est utilisée dans le cadre de la représentation de textes issus d'un corpus de documents.

Analyse statistique : la méthode LSA

La méthode LSA fut brevetée en 1988 aux États-Unis et publiée en 1990 (DEERWESTER et al. 1990). Elle sert à découvrir de manière statistique la sémantique sous-jacente (latente) de mots dans un corpus de documents. Cela va notamment permettre de mettre en relation des requêtes et des documents relevant de celle-ci présents au sein d'un corpus. La méthode se base pour cela sur une matrice de taille $t \times d$ de représentation vectorielle des documents où t est le nombre de termes dans le corpus et d le nombre de documents.

On dénombre quatre étapes principales :

1. La construction de la matrice de représentation vectorielle X
2. La décomposition de la matrice X en deux matrices orthogonales T_0 et D_0 et en une matrice diagonale de valeurs singulières S_0
3. La réduction des matrices S_0, T_0 et D_0 au rang k
4. Le calcul de la matrice $\widehat{X}$, matrice approchée au rang k de la matrice X

Construction de la matrice de représentation vectorielle X La matrice X de taille $t \times d$ des occurrences des termes dans les documents est construite. Chaque ligne de la matrice constitue le vecteur représentatif d'un terme et chaque colonne constitue le vecteur représentatif d'un document. L'intersection de chaque ligne et de chaque colonne représente l'importance du terme représenté par le vecteur ligne dans le document représenté par le vecteur colonne. Cette importance est souvent normalisée en un nombre compris entre 0 (aucune importance) et 1 (extrêmement important). La méthode TF-IDF est souvent employée.

Décomposition de la matrice X en valeurs singulières La matrice X est décomposée en valeurs singulières selon les matrices T_0, S_0, D_0. On a ainsi $X = T_0.S_0.D_0$ avec S_0 matrice diagonale de taille $m \times m$ de valeurs singulières et T_0 et D_0, deux matrices orthogonales de taille $t \times m$ et $m \times d$

Réduction des matrices au rang k La matrice diagonale S_0 de taille $m \times m$ est réduite à une matrice diagonale S de taille $k \times k$ avec $k \leq m$. On effectue une réduction au rang k en ne gardant que les k plus grandes valeurs singulières de la matrice S_0 et en considérant comme nulles les autres valeurs singulières. On appelle S, T et D les matrices ainsi obtenues de tailles respectives $k \times k$, $t \times k$ et $k \times d$.

Calcul de la matrice $\widehat{X}$ La matrice $\widehat{X}$ de taille $t \times d$, approximation au rang k de la matrice X est déduite : $\widehat{X} = T.S.D$

La matrice $\widehat{X}$ ainsi obtenue est une matrice approchée (et donc simplifiée) de la matrice initiale X ne contenant que la sémantique des mots la plus importante d'un point de vue statistique pour le corpus. On peut utiliser cette matrice pour déterminer des distances vectorielles entre deux objets qu'il s'agisse de documents ou de termes.

La distance du cosinus des vecteurs constituant les lignes de la matrice (dans le cas des termes) ou de ceux constituant les colonnes de la matrice (dans le cas des documents) est souvent utilisée. Soit t_i, le vecteur de la i^{e} ligne (resp d_i, la i^{e} colonne) de la matrice $\widehat{X}$ représentative du i^{e} terme (resp. document) et t_j, le vecteur de la j^{e} ligne

(resp d_j, la j^e colonne) de la matrice $\widehat{X}$ représentative du j^e terme (resp. document). On a :

$$\begin{aligned} distance(t_i, t_j) &= Cos(t_i, t_j) \\ distance(d_i, d_j) &= Cos(d_i, d_j) \end{aligned}$$

3.4 Les représentations vectorielles de textes par n-grammes

La représentation vectorielle standard vue dans la section précédente ne tient pas compte de l'ordre des attributs.

Pourtant, l'ordre des attributs peut se révéler un marqueur crucial à prendre en compte pour représenter efficacement et justement la sémantique du texte et ce particulièrement lorsque les attributs sont les mots des textes. Ainsi, les représentations fonctionnant sur un principe de « sac de mots » telles que la représentation vectorielle standard conduisent à établir des similitudes fortes entre des textes s'opposant sur un plan sémantique.

Exemple 3.10

Les vêtements sont de bonne qualité mais vos prix ne sont pas corrects
Vos prix sont corrects mais vos vêtements ne sont pas de bonne qualité

Si l'on supprime les mots creux habituels (les, de, vos), la représentation vectorielle standard des deux textes est {(vêtements, 1), (sont, 2), (bonne, 1), (qualité, 1), (mais, 1), (prix, 1), (ne, 1), (pas, 1), (corrects, 1)}.

Les deux phrases en exemple ont donc la même représentation vectorielle alors qu'elles ont des sens différents.

Afin de contourner cette difficulté, il est usuel d'avoir recours à des représentations prenant en compte l'ordre des attributs de la représentation afin de lever l'ambiguïté provoqué par le sac de mots.

3.4.1 Les représentations n-grammes

Les systèmes n-grammes peuvent être utilisés dans les représentations vectorielles des textes pour prendre en compte l'ordre des attributs.

Définition 3.15 (*n-gramme*)

*Un n-gramme (*SHANNON *1948) est une suite de n symboles linguistiques consécutifs dans le texte à traiter / représenter.*

Dans les travaux de Shannon, le symbole linguistique utilisé est le caractère, la lettre. Dans des travaux plus récents, le mot est parfois utilisé. Nous nous intéressons particulièrement à ces cas.

On désigne par unigramme, un symbole linguistique isolé, bigramme, une suite de deux symboles linguistiques consécutifs et trigramme, une suite de trois symboles consécutifs.

Soit **UG**, la suite d'unigrammes que nous pouvons former à partir des symboles d'un texte et que nous définissons formellement de la manière suivante :

$$\mathbf{UG} = (ug_i)_{[\![1,|UG|]\!]}$$

Dans le cas d'unigrammes, la suite **UG** obtenue est alors équivalente à une suite de mots **M** ou à une suite de caractère **C** selon le symbole linguistique utilisé. Nous avons présenté ces cas particulier au préalable et avons déjà discuté du problème posé par la non-prise en compte de l'ordre des mots ou des caractères.

Nous devons, pour contourner ce problème, augmenter le cardinal des n-grammes de la suite. Pour cela, à partir d'une suite d'unigrammes **UG**, on peut former une suite **NG** de n-grammes de cardinalité n > 1 :

$$\mathbf{NG} = (ng_i)_{[\![1,|NG|]\!]} \text{ avec } ng_i = (ug_j)_{[\![i,i+n-1]\!]}$$

Nous pouvons ainsi définir la représentation vectorielle de n-grammes comme suit :

Définition 3.16 (*représentation vectorielle standard des n-grammes d'un texte*)

La représentation vectorielle standard des n-grammes d'un texte représente un texte constitué d'un ensemble NG de n-grammes ng de cardinalité n distincts par un vecteur tel que la valeur de chaque composante représentative d'un n-gramme ng distinct du texte vaut le nombre d'occurrences du n-gramme ng dans la suite de n-grammes **NG**.
Le nombre d'occurrences d'un n-gramme donné dans le texte est dénoté

$$|ng|_{VSG} = \left|\left\{i \in [\![1, |\mathbf{NG}|]\!] / \ ng_i = ng\right\}\right|$$

La représentation vectorielle standard des n-grammes d'un texte est donnée par le graphe de la fonction $|.|_{VSG}$.

Les deux textes qui avaient une représentation vectorielle standard identique (cf. exemple 3.10) ont cette fois des représentations par 2-grammes de mots différentes.

Exemple 3.11

Les vêtements sont de bonne qualité mais vos prix ne sont pas corrects : {(vêtements, sont), (sont, bonne), (bonne qualité), (qualité, mais), (mais, prix), (prix, ne), (ne, sont), (sont, pas), (pas corrects)}
Vos prix sont corrects mais vos vêtements ne sont pas de bonne qualité : {(prix, sont), (sont, corrects), (corrects, mais), (mais, vêtements), (vêtements, ne), (ne, sont), (sont, pas), (pas, bonne), (bonne, qualité)}

Sur les neuf 2-grammes de mots qui constituent les représentations des deux textes, seuls deux sont communs aux deux représentations. Les textes sont donc cette fois clairement dissociés.

De même, des représentations par 3-grammes de mots peuvent être utilisées.

Exemple 3.12

Les vêtements sont de bonne qualité mais vos prix ne sont pas corrects : {(vêtements, sont, bonne), (sont, bonne, qualité), (bonne qualité, mais), (qualité, mais, prix), (mais, prix, ne), (prix, ne, sont), (ne, sont, pas), (sont, pas, corrects)}
Vos prix sont corrects mais vos vêtements ne sont pas de bonne qualité : {(prix, sont, corrects), (sont, corrects, mais), (corrects, mais, vêtements), (mais, vête-

> *ments, ne), (vêtements, ne, sont), (ne, sont, pas), (sont, pas, bonne), (pas, bonne, qualité)}*

Sur les huit 3-grammes de mots qui constituent les représentations des deux textes, un seul apparaît dans les deux représentations. Les textes sont donc cette fois encore mieux dissociés qu'avec les représentations par 2-grammes de mots.

Dans (VERNIER et al. 2007), une représentation en 3-grammes de mots est utilisée pour effectuer de la classification de textes d'opinions. Cette approche est de plus comparée à une approche lexico-syntaxique. Les résultats obtenus montrent une qualité globalement supérieure de l'approche n-grammes de mots lorsque la taille du corpus de textes est suffisamment grande.

Les n-grammes de mots sont utilisés dans les travaux de (CARLSON et FETTE 2007) et (ISLAM et INKPEN 2009) pour effectuer de la correction orthographique à partir d'un entraînement sur les corpus de n-grammes de mots de Google (FRANZ et BRANTS 2006).

D'une manière générale, les n-grammes de mots permettent de définir la notion de co-occurrences de mots dans un texte : des mots sont co-occurrents dans un texte s'ils apparaissent dans le même n-gramme de mots du texte.

Dans (LUND, BURGESS et ATCHLEY 1995), la méthode *Hyperspace Analogues to Language* (HAL) permet de calculer statistiquement la proximité des mots d'un corpus de documents par leurs co-occurrences dans les n-grammes de mots du corpus.

3.4.2 Analyse statistique : La méthode HAL

La méthode HAL procède en deux phases :

1. Construction de la matrice des co-occurrences X du texte
2. Calcul des vecteurs de co-occurrences du texte

Construction de la matrice des co-occurrences X du texte

La première étape consiste à construire une matrice dite de co-occurrences. Cette matrice est construite à partir des co-occurrences formées par les n-grammes de

mots des textes. Le cardinal des n-grammes est choisi selon une taille de fenêtre à analyser. Une pondération est également couramment appliquée sur cette co-occurrence partant du principe que plus les mots sont proches dans le n-gramme, plus la co-occurrence est importante. Dans (LUND, BURGESS et ATCHLEY 1995), une pondération simple est proposée :

$$P = \text{ taille de la fenêtre } + 1 - \text{ distance entre les deux mots} \tag{3.1}$$

La matrice X de taille $|M| \times |M|$, où $|M|$ correspond à la taille de l'ensemble des mots des textes, est construite à partir des co-occurrences détectées. Lorsque le cardinal est supérieur à 2, une même co-occurrence apparaissant dans plusieurs n-grammes n'est comptabilisée qu'une seule fois. Chaque vecteur ligne de la matrice représente le contexte gauche du mot de la ligne (à savoir le degré avec lequel chaque mot du lexique précède le mot de la ligne) tandis que chaque vecteur colonne représente le contexte droit du mot de la colonne (à savoir le degré avec lequel chaque mot du lexique suit le mot de la colonne).

Calcul des vecteurs de co-occurrences du texte

Il est possible à partir de la matrice de co-occurrences X de déduire le vecteur de co-occurrences de chaque mot des textes. Un vecteur de co-occurrence pour le i^e mot correspond à l'union de la i^e ligne et de la i^e colonne de la matrice (en d'autres termes à l'union des contextes droits et gauches du mot).

Calcul statistique de la proximité des mots

La proximité statistique des mots peut être déduite par calcul de similarité entre les vecteurs de co-occurrences des mots du lexique. La similarité du cosinus est assez couramment employée.

3.5 Les représentations de textes par arbres syntaxiques

Un texte peut être représenté à l'échelle des phrases le constituant. Dans ce cadre, il est possible de compléter la représentation de ce texte en utilisant les relations syntaxiques entre les mots de la phrase en tant qu'informations supplémentaires.

3.5.1 Fondements de l'analyse syntaxique

Définition 3.17 (*Analyse syntaxique*)

L'analyse syntaxique consiste à déterminer pour une phrase d'un texte l'ensemble des relations syntaxiques existantes entre les mots le constituant.

Les relations syntaxiques peuvent être représentées en tant qu'annotations.

Dans certains cas, il n'est pas possible ou pas nécessaire de réaliser une analyse syntaxique dans son intégralité sur les phrases. Une analyse syntaxique de surface peut alors être conduite à la place.

Définition 3.18 (*Analyse syntaxique de surface*)

L'analyse syntaxique de surface (chunking) vise à identifier les liens primaires (les plus simples) existant entre les mots de la phrase.

Afin de réaliser une analyse syntaxique sur les phrases d'un texte, il est nécessaire d'établir au préalable les règles de la syntaxe de la langue utilisée dans le texte. Pour cela, nous utilisons une grammaire.

Définition 3.19 (*Grammaire*)

Une grammaire est un ensemble de règles visant à définir la syntaxe d'une langue.

La complexité des langues rend en fait impossible la constitution d'une grammaire exhaustive concernant les règles de leur syntaxe. Les grammaires constituées visent donc essentiellement à représenter les règles principales de leur syntaxe.

Langage formel	Mot	Alphabet	Symbole
Langage naturel	Phrase	Ensemble des mots	Mot

TABLE 3.1 – Correspondance de termes entre langage formel et langage naturel. Les termes de chaque langage sont données sur la ligne correspondante. Les correspondances entre les langages sont données en colonne.

3.5.2 Langages et grammaires formels

En théorie des langages, des grammaires sont également utilisées. Contrairement au domaine du langage naturel, les langages dits formels sont stricts, simples, non évolutifs et non ambigus.

Certains termes employés pour les langages formels sont « proches » de ceux employés pour les langages naturels. Ils ne sont cependant pas à confondre avec ces derniers.

Définition 3.20 (*Alphabet (en langage formel)*)

Un alphabet $Alpha$, en langage formel, est un ensemble fini de symboles.

Définition 3.21 (*Mot (en langage formel)*)

Un mot, en langage formel, est une suite finie de symboles de l'alphabet $Alpha$.

Définition 3.22 (*Langage formel*)

Un langage formel est un ensemble de mots sur un alphabet $Alpha$.

Le problème de base des langages formels est le problème du mot. Il s'agit de déterminer pour un mot donné si celui-ci appartient ou non au langage formel.

Le tableau 3.1 donne les correspondances entre les notions vues pour les langages naturels et les notions introduites ici pour les langages formels. Le problème du mot appliqué aux langages naturels revient à déterminer l'appartenance d'une phrase à l'ensemble des phrases constituant le langage naturel.

Définition 3.23 (*Grammaire formelle*)

Une grammaire formelle est un formalisme visant à décrire la syntaxe d'un langage formel. Elle est constituée par l'association :

1. *d'un axiome, symbole non terminal de la grammaire, noté conventionnel-*

lement S.

2. *de symboles non terminaux, notés conventionnellement en majuscule*
3. *de symboles terminaux, notés conventionnellement en minuscule*
4. *de règles de production, de la forme $A \rightarrow B$, où A et B sont des suites de symboles non terminaux ou terminaux*

La principale caractéristique des grammaires formelles par rapport aux grammaires utilisées pour les langages naturels est qu'elles sont exhaustives quant à l'ensemble des règles définissant la syntaxe. Dans ces conditions, le lien existant entre la syntaxe du langage et la grammaire est tel que l'on parle de langage généré par une grammaire ou de grammaire générative d'un langage.

Les travaux de Noam Chomsky (CHOMSKY 1956) ont permis d'établir que les différents langages formels et les grammaires formelles associées peuvent être classés selon une hiérarchie couramment désignée par hiérarchie de Chomsky. Celle-ci comporte quatre niveaux :

– Type 0 : Les grammaires générales. Les langages associés sont acceptables par les machines de Turing.

– Type 1 : Les grammaires contextuelles. Les langages associés sont reconnaissables par les automates linéairement bornés.

– Type 2 : Les grammaires non contextuelles. Les langages associés sont reconnaissables par les automates à pile.

– Type 3 : Les grammaires régulières. Les langages associés sont reconnaissables par les automates finis.

Définition 3.24 (*Langage engendré par une grammaire*)

Un langage L est dit engendré par une grammaire formelle si L est l'ensemble des mots construits à partir des symboles terminaux et qui possèdent une dérivation à partir de l'axiome.

Définition 3.25 (*Dérivation de la grammaire*)

Une dérivation de la grammaire est une suite finie $A_1 ... A_n$ de mots constitués de symboles terminaux ou non et telle que pour chaque $i \in 2...m$, il existe une règle

$A \rightarrow B$ telle que $A_{i-1} = CAD$ et $A_i = CBD$ où C et D sont des mots constitués de symboles terminaux ou non.

Définition 3.26 (*Arbre de dérivation*)

Un arbre de dérivation ou arbre syntaxique pour le mot A à partir de l'axiome de la grammaire est un arbre qui synthétise toutes les dérivations de A (elles différent par l'ordre des réécritures).

Les langages naturels n'étant pas des langages formels, il n'est pas possible d'utiliser des grammaires formelles pour les traiter de manière parfaite. Lorsque les textes sont rédigés dans une langue correcte et simple, il est cependant souvent possible de les traiter partiellement avec des grammaires non contextuelles. Lorsque les textes ne sont pas correctement rédigés comme cela est le cas dans nos traitements, une analyse complète n'est en général pas réalisable.

Nous considérons dans les sections suivantes deux types d'analyses qui peuvent être conduites sur des textes :

- Les analyses par constituants
- Les analyses par dépendances

3.5.3 Analyse syntaxique par constituants

Nous présentons dans cette section l'analyse syntaxique par constituants.

Définition 3.27 (*Constituant d'une phrase*)

Un constituant d'une phrase correspond à une sous-partie de la phrase tel qu'il soit possible de substituer cette partie par une autre plus simple ou de complexité équivalente sans perturber la structure grammaticale de la phrase.

Définition 3.28 (*Analyse syntaxique par constituants*)

*L'analyse syntaxique par constituants ou par constituants immédiats (*BLOOMFIELD *1933) analyse la syntaxe d'une phrase en donnant les relations existantes entre les différents constituants de cette dernière. On distingue deux types de constituants : les constituants simples correspondants aux morphèmes libres de la phrase et les constitués correspondants aux constituants formés par agréga-*

Annotation	**Catégorie lexicale**	**Annotation**	**Catégorie lexicale**
A	adjectif	I	interjection
Adv	adverbe	NC	nom commun
CC	conjonction de coordination	NP	nom propre
Cl	pronom clitique faible	P	préposition
CS	conjonction de subordination	PREF	préfixe
D	déterminant	PRO	pronom fort
ET	mot étranger	V	verbe
PONCT	ponctuation		

TABLE 3.2 – Annotations dans le *French Treebank* pour les catégories lexicales. Les annotations sont données dans la 1re et 3e colonne de la table et correspondent respectivement aux catégories lexicales données dans la 2e et 4e colonne de la table et ce pour chaque ligne.

tions successives de constitués et/ou de constituants simples.

Définition 3.29 (*Grammaire de constituants*)

Une grammaire de constituants est une grammaire dont le formalisme se base sur les relations entre constituants pour décrire la syntaxe d'une langue.

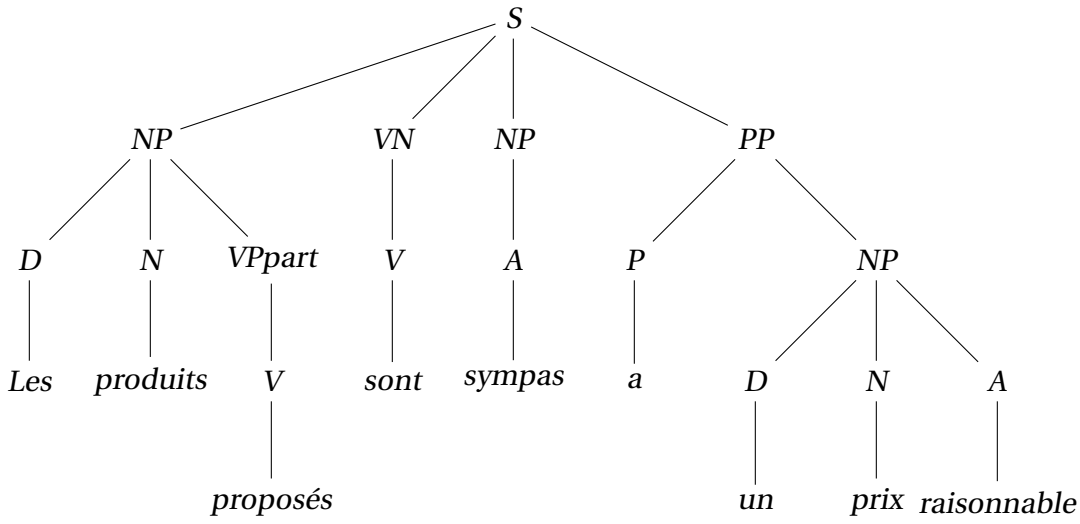
Dans nos exemples, nous utilisons les annotations définies dans le projet *French Treebank* (ABEILLÉ, CLÉMENT et TOUSSENEL 2003). Ce projet distingue 15 catégories lexicales (ou constituants primaires) pour les mots simples (Table 3.2) et 12 annotations pour les constitués (Table 3.3).

Annotation	Constitué	Annotation	Constitué
AP	phrases adjectivales	VN	noyau verbal
AdP	phrases adverbiales	VPinf	clauses infinitives
COORD	phrases coordonnées	VPpart	clauses non finies
NP	phrases nominales	SENT	phrases
PP	phrases prépositionelles	Sint, Srel, Ssub	clauses finies

TABLE 3.3 – Annotations dans le *French Treebank* pour les constitués. Les annotations sont données dans la 1re et 3e colonne de la table et correspondent respectivement aux constitués donnés dans la 2e et 4e colonne de la table et ce pour chaque ligne.

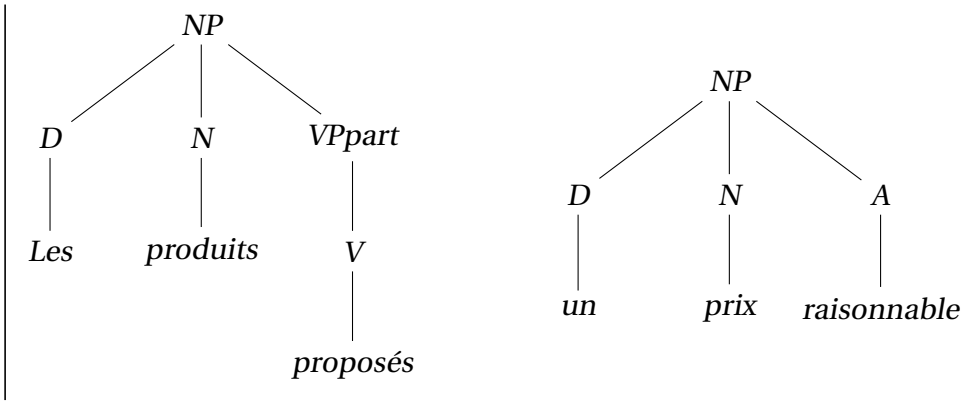
Exemple 3.13

L'analyse syntaxique obtenue sur la phrase « Les produits proposés sont sympas à un prix raisonnable » par une analyse par constituants est la suivante :



Exemple 3.14

L'analyse syntaxique de surface, orientée sur la détection des groupes nominaux, produit sur la phrase « Les produits proposés sont sympas à un prix raisonnable » le résultat suivant :



Dans le cas d'une analyse de surface, l'arbre syntaxique obtenu est partiel puisqu'il ne va pas jusqu'à l'axiome de la grammaire mais regroupe simplement certains symboles terminaux de la suite en symboles non terminaux plus complexes.

Les algorithmes Earley (EARLEY 1970) et CYK (KASAMI 1965) (YOUNGER 1967) (COCKE 1969) sont des algorithmes d'analyse syntaxique très répandus dans le cadre de grammaire non contextuelles. Leur approche du problème est différente et reflète les deux techniques d'approche existantes :

1. L'approche descendante qui s'effectue depuis l'axiome et essaye par application des règles d'arriver à l'instance fournie. C'est la technique retenue par l'algorithme Earley.
2. L'approche ascendante qui s'effectue depuis l'instance et essaye par rétro application des règles de production de revenir à l'axiome. C'est la technique retenue par l'algorithme CYK.

Ces deux approches différent essentiellement par le sens dans lequel elles parcourent les règles de la grammaire pour vérifier leur adéquation avec la suite des symboles terminaux présentés en entrée.

L'approche descendante part de l'axiome et descend dans la grammaire jusqu'à reconnaître les terminaux présentés en entrée. Si la suite est trop bruitée, l'analyse peut échouer très tôt et ce sans avoir reconnu un seul symbole terminal. Lorsque ce cas ce produit, l'analyse est totalement infructueuse.

A contrario, l'approche ascendante débute son analyse par une analyse syntaxique de surface pour continuer ensuite sur une analyse plus profonde de la syntaxe de la

phrase. L'avantage de cette approche est de garantir le retour d'un résultat pourvu que l'analyse de surface réussisse au moins partiellement. L'analyse obtenue est alors qualifiée de partielle. Dans le cadre de notre exemple, l'échec de l'analyse peut ainsi résulter en le renvoi uniquement des deux groupes nominaux (cas où l'analyse de surface a entièrement réussi) ou sur seulement l'un des deux.

3.5.4 Analyse syntaxique par dépendances

Nous présentons dans cette section l'analyse syntaxique par dépendances.

Définition 3.30 (*Analyse syntaxique par dépendances*)

*L'analyse syntaxique par dépendances (*TESNIERE *1959) analyse la structure des phrases en dégageant les relations de rôle existant entre les différents éléments de la phrase.*

Par exemple, le sujet peut voir sa relation avec le verbe explicitée par un rôle d'acteur de l'action exercée sur le complément d'objet direct qui a alors un rôle d'agent par rapport au verbe subissant l'action. Donnons quelques définitions formelles utiles pour mieux expliciter ce type d'analyse.

Définition 3.31 (*Élément régisseur, élément subordonné*)

Un élément régisseur est un élément qui en analyse par dépendances gouverne la dépendance. C'est donc l'élément principal de la dépendance. Un élément subordonné est un élément qui en analyse par dépendances subit la dépendance. C'est donc l'élément secondaire de la dépendance.

Comme pour l'analyse syntaxique par constituants, l'analyse syntaxique peut être représentée de manière formelle en utilisant des grammaires.

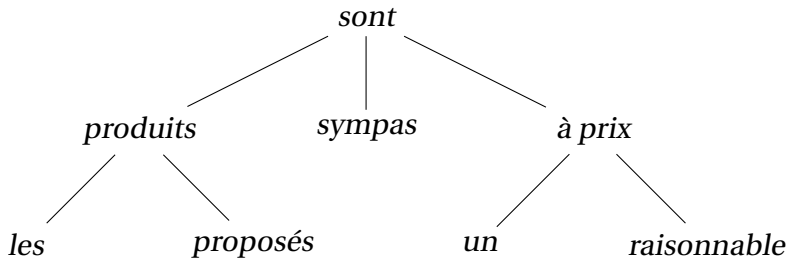
Définition 3.32 (*Grammaire de dépendances*)

Une grammaire de dépendances est une grammaire dont le formalisme se base sur les relations de dépendances pour décrire la syntaxe d'une langue.

Bien que le terme employé soit dans les deux cas celui de grammaire, une grammaire de constituants est différente d'une grammaire de dépendances.

Exemple 3.15

Cet exemple donne l'arbre syntaxique obtenu sur la phrase « Les produits proposés sont sympas à un prix raisonnable » par une analyse par dépendances.



L'élément central de la phrase est le verbe définissant l'action d'être. A partir de cette action régissante, nous avons 3 dépendances avec des éléments subordonnés :

1. *Le subordonné primaire correspond à l'agent sujet (ou l'agent acteur) de l'action : « produits » ;*
2. *Le subordonné secondaire correspond à l'agent objet (ou l'agent subissant) l'action : « sympas » ;*
3. *Le subordonné tertiaire correspond à un complément indirect de l'action : « à prix ».*

Chacun des ces éléments subordonnés du verbe sont eux-même éléments régisseurs d'autres éléments subordonnés.

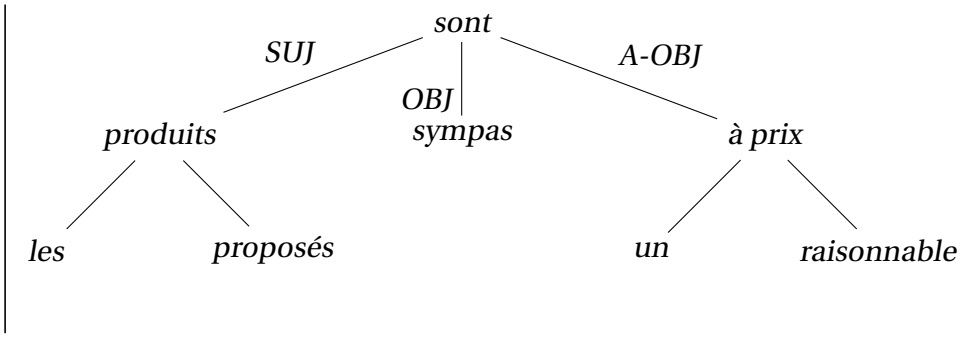
Il est possible d'utiliser une représentation par arbre syntaxique explicitant directement les rôles de subordination des éléments entre eux. Nous utilisons encore une fois dans nos exemples, les 8 annotations définies pour le projet *French Treebank* (ABEILLÉ, CLÉMENT et TOUSSENEL 2003) (Table 3.4).

Exemple 3.16

Cet exemple donne l'arbre syntaxique obtenu sur la phrase « Les produits proposés sont sympas à un prix raisonnable » par une analyse par dépendances avec les rôles explicites.

Annotation	**Relation de dépendance**	**Annotation**	**Relation de dépendance**
SUJ	sujet	MOD	modificateur ou complément
OBJ	objet direct	A-OBJ	complément indirect introduit par à
ATS	complément prédicatif du sujet	DE-OBJ	complément indirect introduit par de
ATO	complément prédicatif de l'objet direct	P-OBJ	complément indirect introduit par une autre préposition

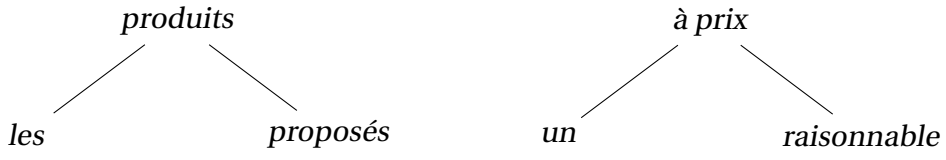
TABLE 3.4 – Annotations dans le *French Treebank* pour les relations de dépendance. Les annotations sont données dans la 1re et 3e colonne de la table et correspondent respectivement aux relations de dépendance données dans la 2e et 4e colonne de la table et ce pour chaque ligne.



De même que pour l'analyse par constituants, il n'est pas toujours possible ou nécessaire de réaliser l'analyse par dépendances dans son intégralité. Dans ce cas, une analyse de surface peut être réalisée.

Exemple 3.17

Cet exemple donne l'arbre syntaxique obtenu sur la phrase « Les produits proposés sont sympas à un prix raisonnable » par une analyse de surface par dépendances.



3.5.5 Les corpus arborés : apprentissage artificiel des grammaires

Des grammaires peuvent être générées de manière manuelle quand cela ne constitue pas une tâche trop coûteuse. Malheureusement, si l'on désire une grammaire suffisamment exhaustive des règles syntaxiques d'une langue, cela n'est pas le cas.

Dans ce cas, les méthodes d'apprentissage artificiel peuvent être utilisées pour contourner cette difficulté en introduisant une part d'automatisme dans la construction de la grammaire, la rendant par là même semi-automatique. Nous discutons plus en détail de ce qu'est un apprentissage artificiel au chapitre 6.

Pour apprendre des règles de grammaire, on utilise un corpus arboré :

Définition 3.33 (*Corpus arboré (Treebank)*)

Un corpus arboré (Treebank) est un corpus annoté morphologiquement et syn-

taxiquement.

Ces *corpora* font la plupart du temps l'objet de validations manuelles afin de garantir une certaine qualité des annotations proposées. Ils sont souvent utilisés par des algorithmes d'apprentissage supervisé dans le cadre d'un apprentissage semi-automatique d'une grammaire. Les textes annotés de ces *corpora* servent alors de données d'entraînement et tests pour ces algorithmes.

Dans le cadre des langues, divers projets d'élaboration de corpus arborés ont été initiés pour couvrir divers domaines et diverses langues. Parmi eux, nous citerons :

1. le *Penn Treebank* (MARCUS, MARCINKIEWICZ et SANTORINI 1993) construit à partir des quatre corpus : *Wall Street Journal*, *The Brown Corpus*, *Switchboard* et ATIS. Il a été construit par le laboratoire LINC de l'université de Pennsylvanie. Ce corpus fait référence dans le traitement de la langue anglaise.
2. le *French Treebank* (ABEILLÉ, CLÉMENT et TOUSSENEL 2003) construit à partir du corpus du journal Le Monde, extrait entre décembre 1989 et janvier 1994. Il a été construit par le laboratoire LLF de l'université Paris 7. Ce corpus fait référence dans la langue française.
3. le corpus Séquoia (CANDITO et SEDDAH 2012) construit à partir de quatre corpus issus de l'agence européenne du médicament, Europarl, le journal régional l'Est Républicain et Wikipedia Fr. Il a été construit par l'équipe Alpage de l'université Paris 7 et l'Inria.

Il est à noter que le dernier corpus arboré correspond à un projet récent visant à mettre à disposition un corpus arboré libre pour le français. De plus, le choix de *corpora* sources s'est effectué avec un souci de diversification contrairement à ses deux homologues construits principalement sur des *corpora* de journaux. Cela doit permettre un apprentissage plus robuste au changement de domaine entre l'apprentissage et l'exploitation. Cet absence de robustesse au changement de domaine est particulièrement saillante pour le *Penn TreeBank* : le *Wall Street Journal* qui représente sa principale source de textes montre peu de variété de thèmes (CANDITO et SEDDAH 2012).

Pour exploiter ces *corpora* arborés, des travaux de recherche ont été conduits dans le but de développer des algorithmes d'apprentissage artificiel pour apprendre

des grammaires. Nous citerons en exemple :

1. Le parseur de Berkeley (PETROV, BARRETT et al. 2006) (PETROV et D. KLEIN 2007) qui permet d'analyser syntaxiquement les phrases par constituants.
2. Le *MaltParser* (NIVRE 2003) (NIVRE, HALL et NILSSON 2004) qui permet d'analyser syntaxiquement les phrases par dépendances.

Le parseur de Berkeley est un parseur syntaxique conçu dans le but d'apprendre de manière supervisée une grammaire non contextuelle. Cet apprentissage s'effectue via un corpus arboré tel que ceux présentés ci-dessus, étiquetés pour l'analyse en constituants. Le parseur de Berkeley a notamment été entraîné pour l'anglais sur le corpus arboré *Penn Treebank* (PETROV, BARRETT et al. 2006) pour lequel il a été conçu. Il fut par la suite testé avec succès pour le français sur le *French Treebank* (CRABBÉ, CANDITO et al. 2008). Il a également été entraîné sur du bulgare, du chinois, de l'arabe et de l'allemand. Le parseur ainsi que les modèles obtenus à la suite de ces entraînements sont disponibles sur le site du projet [2].

Le *MaltParser* est un parseur syntaxique de dépendances guidé par les données. L'apprentissage s'effectue également via un corpus arboré mais étiqueté cette fois en dépendances. Le *MaltParser* a notamment été entraîné pour l'anglais sur le *corpus* arboré *Penn Treebank* pour lequel il a été conçu. Il fut par la suite testé avec succès en français sur des *corpora* dérivés du *French Treebank* (CANDITO, NIVRE et al. 2010) et en suédois sur le corpus arboré *Swedish Treebank*. Le parseur ainsi que les modèles obtenus à la suite de ces entraînements sont disponibles sur le site du projet [3].

Dans (CANDITO, NIVRE et al. 2010), le parseur de Berkeley, le *MaltParser* et le *MSTParser* sont évalués et leurs résultats comparés. Les résultats obtenus par les trois analyseurs sont presque identiques en terme de précision générale d'analyse bien que le *MSTParser* semble disposer d'une courte avance sur ces deux challengers. L'écart entre les analyseurs devient en revanche évident en terme de temps de traitement. Celui-ci a été mesuré sur l'analyse du jeu de développement. Le plus long est le *MSTParser* qui prend un peu plus de 14 minutes, suivi du parseur de Berkeley qui prend un peu plus de 12 minutes. Le *MaltParser* se distingue alors en ne prenant

2. `http://code.google.com/p/berkeleyparser/`
3. `http://www.maltparser.org/`

pour sa part qu'une seule minute, soit 12 fois moins de temps que le parseur Berkeley et 14 fois moins de temps que le *MSTParseur*. Ces différences de temps de traitements résident dans la complexité de l'algorithme utilisé : linéaire pour le *MaltParser* et cubique pour les deux autres.

3.6 Conclusion

Dans ce chapitre, nous avons présenté des techniques de représentation des textes. Nous avons d'abord considéré les plus simples (les suites lettres et de mots), utilisées par défaut dans de nombreuses applications.

Nous avons ensuite montré que ces représentations sont également les plus pauvres d'un point de vue des informations linguistiques et sémantiques qui sont représentées. Nous nous sommes alors intéressés à celles issues de la représentation vectorielle standard. Cette famille de représentations présente l'avantage de pouvoir inclure des notions linguistiques et sémantiques au travers de la représentation des concepts de la phrase.

Par ailleurs, la famille des représentations dites sémantiques permet également de représenter des aspects sémantiques des textes en incluant, en plus des concepts, les liens existants entre ceux-ci. Les graphes sémantiques (RASTIER 1989) (SOWA 1984) constituent un exemple de représentation sémantique de textes. Cependant, la complexité de construction de ce type de représentations dans un contexte syntaxiquement incorrect ne nous permet pas d'envisager leur utilisation. De plus, ces représentations ne présentent pas l'avantage de former un espace de type vectoriel, utile pour établir des relations de similarité entre textes comme nous l'avons précisé au chapitre 2. C'est pourquoi nous avons fait le choix de ne pas décrire ce type de représentations dans ce chapitre.

Chapitre 4

Correction orthographique

Les textes que nous avons à traiter ne peuvent être considérés comme orthographiquement corrects. L'utilisation de méthodes de correction orthographique permet d'éliminer (en partie) ce « bruit », ce qui conduit à améliorer les performances des approches utilisant ces textes (par exemple, pour mesurer des similarités entre phrases). Ce chapitre est consacrée à la présentation de différentes méthodes existantes.

4.1 Formalisation du problème et discussions préliminaires

Le processus de correction orthographique est un processus visant à remplacer certains mots des textes par des mots issus d'un lexique constitué des mots de la langue. Les mots à remplacer sont supposés coïncider avec ceux n'appartenant pas au lexique. Tout mot inconnu doit être remplacé par un mot appartenant à ce lexique et ayant des similarités fortes avec le mot d'origine.

Dans le chapitre 2, nous avons introduit quelques notions sur le calcul de similarités dans des espaces métriques. Une approche en correction orthographique est de concevoir un espace métrique sur les mots reflétant leur proximité orthographique.

Cette proximité n'est bien sûr pas évidente à déterminer et constitue la difficulté majeure dans la conception d'un espace métrique idéal.

Posons à présent une définition plus formelle de la correction orthographique.

Définition 4.1 (*Correction orthographique*)

Soient L l'ensemble des lexèmes de la langue et M l'ensemble des mots constituables à partir de suites de caractères représentatifs des symboles de la langue. La correction orthographique est le nom donné à la relation $\sim_c$, de graphe G_c, visant à associer un mot $m \in M$ à des lexèmes $l \in L$.

$$G_c \subseteq M \times L$$

Notons que nous n'avons pas donné dans notre définition de propriétés permettant de préférer l'association d'un mot $m \in M$ avec un mot $l \in L$ plutôt qu'un mot $l' \in L$. Formellement, nous cherchons alors à établir une relation d'ordre sur les éléments de L selon le mot $m \in M$ à corriger par la relation $\sim_c$.

Une manière simple de définir cette relation d'ordre est d'utiliser une similarité sim de M^2 dans $\mathbb{R}^+$ vérifiant les propriétés :

Séparation des lexèmes : $sim(m,l) = 1 \Leftrightarrow l = m$
Unicité de la similarité : $\forall m \in M, \forall (l,l') \in L^2, sim(m,l) = sim(m,l') \Rightarrow l = l'$

La relation de correction s'exprime alors par :

$$m \sim_c l \Leftrightarrow \forall l' \in L, sim(m,l') \leq sim(m,l)$$

Cette relation de correction possède les propriétés suivantes :
Non correction des lexèmes : $m \sim_c m \Leftrightarrow m \in L$
Unicité de la correction : $\forall m \in M, \forall (l,l') \in L^2, m \sim_c l \wedge m \sim_c l' \Rightarrow l = l'$
Idempotence de la correction : $\forall m \in M, \forall (l,l') \in L^2, m \sim_c l \wedge l \sim_c l' \Rightarrow l = l'$

Il est bien sûr possible d'utiliser une dissimilarité, un indice de distance ou une distance à la place de la similarité à condition d'adapter la dernière propriété. La

relation de correction s'exprime alors :

$$m \sim_c l \Leftrightarrow \forall l' \in L, diss(m, l') \geq diss(m, l)$$

La correction orthographique se confronte à trois contraintes/difficultés fortes.

La première concerne la définition d'un mot erroné. La correction orthographique se basant sur la non-appartenance du mot au lexique de la langue, il n'est pas possible de considérer comme erroné un mot du lexique ne devant normalement pas apparaître dans la phrase.

Exemple 4.1

> *produits bien mais **chair** (surtout la déco ! !)*

Exemple 4.2

> *très bonne qualité des articles en général, et correspondance des tailles...Les tissus sont de bonne **existence** aux lavages... J'ai assez rarement été déçue...*

Le mot « chair » appartient bien au lexique de la langue mais n'est pas correct dans la phrase (substitution de « cher »). La correction orthographique le laissera inchangé. Il en est de même avec le mot « existence » qui correspond ici à une substitution de « résistance ».

Afin de détecter ce type d'erreurs, nous devons avoir recours à des corrections syntaxico-sémantiques prenant en compte à la fois la syntaxe attendu de la langue et le sens des textes pour remplacer les mots erronés par ceux corrects. Étant donné le contexte syntaxique peu rigoureux des textes que nous avons à traiter, il nous semble peu judicieux de réaliser une analyse aussi complexe.

La deuxième difficulté est également liée à la notion de lexique des mots de la langue mais cette fois dans le cadre d'un mot correct inconnu. Si un mot correct n'appartient pas au lexique, la correction orthographique va chercher à le remplacer à tort par un autre mot de la langue.

Exemple 4.3

> *Ce que j'apprécie particulièrement, c'est que je peux trouver des vêtements simples et classiques, sans dessins superflus et autres personnages commerciaux (Dora,*

Barbie, Spiderman...)

Dora, Barbie et Spiderman étant des noms de personnages, ils n'appartiennent certainement pas au lexique traditionnel de la langue. La correction orthographique pourrait alors vouloir les remplacer par d'autres mots comme « barbue », « barbier » ou « épidermes » qui ne sont bien sûr pas du tout corrects ici. Sur ce point, la plupart des correcteurs orthographiques proposent l'ajout de mots dans le lexique afin de contourner cette difficulté.

La troisième contrainte concerne les notions de similarités, distances ou dissimilarités calculées entre les morphèmes libres et les lexèmes de la langue. Les mesures de proximité entre mots dans l'espace métrique choisi doivent refléter au maximum la proximité perçue entre les morphèmes libres n'appartenant pas au lexique et les lexèmes leur correspondant d'un point de vue orthographique.

Exemple 4.4

Pas d'attete, des prix clairs une réduction importante

Dans cette exemple, la mesure utilisée doit être capable de déterminer une similitude plus forte entre les mots « attete » et « attente » qu'entre « attete » et n'importe quel autre mot du lexique pour que la correction orthographique puisse fonctionner. Comme dans le cas des textes vus au chapitre 3, il est intéressant d'utiliser une représentation des mots conduisant à un espace métrique. La section 4.2 présente les moyens usuels de construire ces représentations. La section 4.3 est consacrée à l'étude plus approfondie des mesures à employer dans les espaces métriques de représentations des mots. Néanmoins, cette unicité proposée de la correction repose sur la capacité de la similarité à toujours proposer un unique meilleur candidat, ce qui ne peut être le cas que lorsque celle-ci s'appuie sur le sens des mots. Prenons l'exemple du mot à corriger « essance ». 2 candidats sont equi-possibles sans la prise en compte de la sémantique : « essence » (extrait de végétaux) et « essence » (carburant). Ce problème peut être usuellement contourné en effectuant la correction orthographique non pas dans l'ensemble des lexèmes L mais dans le sous-ensemble de L constitué des mots orthographiquement distincts. Nous optons pour cette stratégie dans nos travaux.

4.2 Représentation des mots

Le but des représentations présentées ici est de mettre en évidence les caractéristiques langagières des mots. Il est usuel de distinguer deux types de représentations en fonction du modèle utilisé :

1. Les représentations de mots dont le modèle est basé sur les caractères alphabétiques ;
2. Les représentations de mots dont le modèle est basé sur les phonèmes.

Ces deux types de représentations sont complémentaires. Les mots « vert » et « verre » sont identiques pour une représentation basée sur les phonèmes (prononciation identique) mais très différents pour une représentation basée sur les caractères. A l'inverse, les mots « loin » et « lion » sont peu éloignés pour une représentation basée sur les caractères (inversion des caractères 'o' et 'i') mais très éloignés pour une représentation par phonèmes (perte du son 'oin' et apparition des sons 'i' et 'on').

4.2.1 Ensemble des mots

Intéressons-nous d'abord à l'ensemble des mots construit à partir d'une suite de caractères alphabétiques. Cet ensemble étant infini, il n'est pas facile à étudier. Il est de ce fait usuel d'étudier son sous-ensemble constitué uniquement des morphèmes libres constituant des mots réels de la langue. Nous nous intéresserons pour notre part principalement aux langues anglaises et françaises et donc aux deux ensembles constitués par les mots de ces deux langues.

Quels sont les ordres de grandeur de ces deux ensembles ? Une première étude dans ce sens fondée sur les dictionnaires Larousse et Cambridge donne les chiffres suivants :

* 135000 représentations [1] distinctes en français.
* 35000 représentations [2] distinctes en anglais.

1. Nombre de définitions dans le dictionnaire Larousse au 27/05/2012 `http://www.larousse.fr/dictionnaires/francais`
2. Nombre de définitions dans le dictionnaire Cambridge Business au 27/05/2012 `http://www.cambridge.org/fr/elt/catalogue/subject/project/item6595771/Cambridge-Business-English-Dictionary`

Cependant, les dictionnaires comme le Larousse ou le Cambridge ne donnent pas l'ensemble des lexèmes constituant la langue mais uniquement ceux considérés comme des lemmes. A l'inverse, des dictionnaires collaboratifs en ligne bien que certainement non exhaustifs donnent également les lexèmes flexionnels. Une deuxième étude portant sur le Wiktionnaire donne les chiffres suivants :

* 1178981 représentations [3] distinctes en français.

* 440042 représentations [4] distinctes en anglais.

Soit, en moyenne, 10 fois plus de formes flexionnelles que de formes lemmatisées.

La taille importante de cet ensemble rend difficile son exploitation pour la correction orthographique. Il est donc usuel d'utiliser des représentations visant à simplifier cette exploitation.

Remarque 4. Dans ce chapitre, nous utilisons les exemples filés des mots « chair », « cher », « char », « cri », « cru », « satisfait », « satifait », « prix » et « rpix ».

4.2.2 Représentation des mots par suites finies de caractères

La représentation la plus triviale d'un mot est celle basée sur la suite des caractères le constituant.

Définition 4.2 (*Représentation d'un mot par suite finie de caractères*)

Un mot m constitué de caractères alphabétiques c de la langue peut être représenté par une suite finie de caractères.

$$m = (c_i)_{[\![1,|m|]\!]}$$

3. Nombre de pages en français dans le dictionnaire Wiktionnaire français au 28/05/2012 `http://fr.wiktionary.org/wiki/Cat%C3%A9gorie:fran%C3%A7ais`

4. Nombre d'entrées en anglais dans le dictionnaire Wiktionnaire anglais au 28/05/2012 `http://en.wiktionary.org/wiki/Wiktionary:Statistics`

Exemple 4.5

Les mots « chair », « cher », « char », « cri », « cru », « satisfait », « satifait », « prix » et « rpix » ont les représentations par suites finies de caractères suivantes :

- *(c, h, a, i, r)*
- *(c, h, e, r)*
- *(c, h, a, r)*

- *(c, r, i)*
- *(c, r, u)*
- *(s, a, t, i, s, f, a, i, t)*
- *(s, a, t, i, f, a, i, t)*
- *(p, r, i, x)*
- *(r, p, i, x)*

Cette représentation possède une capacité de distinction des représentations très grande. Seuls les mots homonymes seront confondus par cette représentation.

Exemple 4.6

Les mots homonymes « cher » (tarif élevé) et « cher » (formule de politesse) ont la même représentation par suite finie de caractères : (c, h, e, r)

Cette capacité de distinction forte engendre des représentations distinctes de mots proches en écriture et donc facilement confondables tels que « char » et « cher » ou même « tache » et « tâche ». Il est dès lors nécessaire d'utiliser une distance efficace pour trouver des rapprochements.

4.2.3 Représentation phonétique des mots par la famille des méthodes Soundex

La représentation Soundex

La méthode de représentation phonétique Soundex a été élaborée à l'origine pour la langue anglaise (ODELL et RUSSELL 1918/1922). Elle a été conçue dans l'objectif de permettre une indexation des noms propres par leur phonétique même si elle est aujourd'hui plus largement utilisée sur l'ensemble des mots. Malgré son ancienneté de près d'un siècle, l'algorithme est toujours réputé pour l'indexation dans la langue anglaise et sert notamment dans les implémentations des systèmes de gestion de bases de données.

L'algorithme s'appuie sur deux temps distincts pour obtenir la représentation phonétique :

– Partie 1 : Représentation consonantique du mot

– Partie 2 : Encodage alphanumérique de la représentation consonantique

Le succès de la méthode sur la langue anglaise a conduit à des travaux permettant son adaptation à d'autres langues dont le français.

L'algorithme procède selon six étapes décrites en annexe C.1.1 page C-1 depuis le mot d'origine pour trouver sa représentation Soundex.

Prenons un exemple pour illustrer cette méthode de représentation.

Exemple 4.7

Dans la version pour la langue française de l'algorithme, les mots 'chair', 'cher', 'char', 'cri' et 'cru' ont tous le même code Soundex : 'C600'.

Étape	*Code mot 1*	*Code mot 2*	*Code mot 3*	*Code mot 4*	*Code mot 5*
1	*CHAIR*	*CHER*	*CHAR*	*CRI*	*CRU*
2	*C HAIR*	*C HER*	*C HAR*	*C RI*	*C RU*
3	*C R*	*C R*	*C R*	*C R*	*C R*
4	*C 6*	*C 6*	*C 6*	*C 6*	*C 6*
5	*C 6*	*C 6*	*C 6*	*C 6*	*C 6*
6	*C600*	*C600*	*C600*	*C600*	*C600*

Les mots « satisfait », « satifait », « prix » et « rpix » ont des codes Soundex différents.

Étape	*Code mot 1*	*Code mot 2*	*Code mot 3*	*Code mot 4*
1	*SATISFAIT*	*SATIFAIT*	*PRIX*	*RPIX*
2	*S ATISFAIT*	*S ATIFAIT*	*P RIX*	*R PIX*
3	*S TSFT*	*S TFT*	*P RX*	*R PX*
4	*S 3893*	*S 393*	*P 68*	*R 18*
5	*S 3893*	*S 393*	*P 68*	*R 18*
6	*S389*	*S393*	*P680*	*R180*

Au travers de cet exemple, nous observons les deux principaux problèmes posés par cette représentation.

Premièrement, la très (trop) drastique réduction de la capacité de représentation des mots par rapport au nombre d'éléments originellement distincts dans l'espace

de représentation des mots. En effet, la méthode Soundex réduit l'espace de représentation des mots à :

* $(26 \times 9 \times 9 \times 10) + (26 \times 9 \times 1 \times 1) + (26 \times 1 \times 1 \times 1) = 21060 + 234 + 26 = 21320$ représentations distinctes possibles en français
* $(26 \times 6 \times 6 \times 7) + (26 \times 6 \times 1 \times 1) + (26 \times 1 \times 1 \times 1) = 6552 + 156 + 26 = 6734$ représentations distinctes possibles en anglais

Ces chiffres sont à comparer au nombre d'éléments originellement distincts dans l'espace de représentation des mots. La capacité de discrimination de la méthode Soundex est alors :

* de 2.2% par rapport au nombre d'éléments originellement distincts dans l'espace de représentation des mots français.
* de 2.03% par rapport au nombre d'éléments originellement distincts dans l'espace de représentation des mots anglais.

Le nombre de mots distinguables est donc très sensiblement réduit par la représentation Soundex.

Deuxièmement, il est très difficile d'établir une distance entre des représentations reflétant une proximité linguistique entre les mots. Ainsi les mots « rpix » et « prix » ont des codes assez différents malgré leur proximité alors que des mots très différents comme « char » et « cri » sont à une distance nulle puisque partageant la même représentation.

La représentation Soundex 2

Une méthode dérivée de Soundex visant à l'améliorer a vu le jour en 1995 : la méthode Soundex 2 (CELKO 1995). Elle a été développée par Joe Celko pour la langue anglaise et fut par la suite adaptée pour d'autres langues que l'anglais dont le français (BROUARD 2004).

Comme son prédécessuer, l'algorithme s'appuie sur deux temps distincts pour obtenir la représentation phonétique :

– Partie 1 : Représentation consonantique du mot
– Partie 2 : Encodage alphanumérique de la représentation consonantique

La version pour le français se compose des 14 étapes énumérées en annexe C.1.2 page C-3.

Exemple 4.8

Dans la version pour la langue française de l'algorithme, les mots 'chair', 'cher' et 'char' ont le même code Soundex 2 : 'CHR0'. Les mots 'cri' et 'cru' ont également le même code Soundex 2 mais légèrement différent de celui des autres mots : 'CR00'

Étape	*Code mot 1*	*Code mot 2*	*Code mot 3*	*Code mot 4*	*Code mot 5*
1	*chair*	*cher*	*char*	*cri*	*cru*
2-5	*CHAIR*	*CHER*	*CHAR*	*CRI*	*CRU*
6-10	*CHAAR*	*CHAR*	*CHAR*	*CRA*	*CRA*
11	*CHAAR*	*CHAR*	*CHAR*	*CR*	*CR*
12-13	*CHR*	*CHR*	*CHR*	*CR*	*CR*
14	*CHR0*	*CHR0*	*CHR0*	*CR00*	*CR00*

Les mots « satisfait », « satifait », « prix » et « rpix » ont les codes Soundex 2 suivants :

Étape	*Code mot 1*	*Code mot 2*	*Code mot 3*	*Code mot 4*
1	*satisfait*	*satifait*	*prix*	*rpix*
2-5	*SATISFAIT*	*SATIFAIT*	*PRIX*	*RPIX*
6-10	*SATASFAAT*	*SATAFAAT*	*PRAX*	*RPAX*
11	*SATASFAA*	*SATAFAA*	*PRAX*	*RPAX*
12-13	*STSF*	*STF*	*PRX*	*RPX*
14	*STSF*	*STF0*	*PRX0*	*RPX0*

Nous constatons sur cet exemple que le problème de faible discrimination de Soundex est en partie surmonté par Soundex 2. Même si la représentation Soundex 2 s'effectue toujours sur 4 caractères comme pour la méthode Soundex, l'espace constitué dénombre à présent jusqu'à environ $25 * 20 * 20 * 20 = 200000$ représentations distinctes possibles pour la langue française. La capacité de discrimination de la méthode Soundex 2 est donc de 17% par rapport au nombre d'éléments distincts dans l'espace de représentation des mots.

Cependant, il est toujours très difficile d'établir une distance entre des représentations reflétant une proximité linguistique entre les mots. Nous pouvons observer

que les mots « rpix » et « prix » ont encore des codes assez différents malgré leur proximité.

Utilisation pour le traitement des textos

Au contraire de la représentation par suite finie de caractères, les représentations de la famille Soundex ne possèdent pas une capacité de distinction forte. Cela engendre qu'elles assimilent dans leur représentations les mots proches et donc confondables. Une distance simple, peu efficace dans le cas contraire, telle qu'une correspondance exacte des représentations, est dès lors suffisante à établir des rapprochements pertinents.

Si comme nous l'avons vu, la famille des représentations Soundex n'est que peu pertinente pour le traitement phonétique des langues en écriture usuelle, elle se révèle plus intéressante dans le traitement phonétique des langues en écriture texto.

La représentation des mots par leur squelette consonantique permet en effet de résoudre des alignements de mots écrits en texto avec leurs correspondants écrits normalement (FAIRON, J. KLEIN et PAUMIER 2006). Le tableau 4.1 présente des mots en écriture texto ainsi que leur correspondant en écriture usuelle accompagnés de leur codes Soundex et Soundex 2. La plupart des abréviations textos ont les mêmes codes Soundex et Soundex 2 que leurs homologues en écriture usuelle.

Des problèmes sont relevés dans le traitement des sons voyelliques comportant des caractères consonantiques. Ces problèmes se produisent notamment avec « dans » et « commande » où le son voyellique 'an' n'est pas correctement géré. Ce problème vient de l'origine anglo-saxonne de la méthode n'ayant pas à traiter ce type de cas. Les adaptations à la langue française n'ont pas pris en compte cette divergence.

Mot	Code Soundex	Code Soundex 2
tjrs toujours	T768	TJR0
tt tout	T300	T000
slt salut	S430	SL00
ds dans	D800 D580	D000 DN00
cmd commande cmand	C530 C553	CM00 KMND CMN0

TABLE 4.1 – Codes Soundex et Soundex 2 de mots du français courant et de mots de textos français. Pour chaque ligne, les codes Soundex et Soundex 2 sont donnés dans la 2e et 3e colonne pour le mot de la 1re colonne.

4.2.4 Représentation phonétique des mots par la famille des méthodes Metaphone

La représentation Metaphone

La méthode Metaphone (PHILIPS 1990) a été développée uniquement pour la langue anglaise et son utilisation est de ce fait peu courante sur d'autres langues dont le français.

Cette absence de gestion du français nous conduit à ne pas détailler cette méthode.

La représentation double Metaphone

Fort du succès de l'algorithme de première génération sur la langue anglaise, Lawrence Philips propose en 2000 une amélioration de son premier algorithme Metaphone : double Metaphone (PHILIPS 2000). Cette deuxième génération permet de prendre en compte les spécificités d'autres langues que l'anglais dont les langues slaves, germaniques, grec, français, italien, espagnol et chinois. Malheureusement, ces spécificités sont intégrées dans un seul et même algorithme sans distinction de

langue. Cela nous paraît peu justifié et entraîne des phonétisations inattendues. Par exemple, les mots « thé » (français) et « *the* » (anglais) ont la même représentation phonétique.

Cette absence de séparation dans la gestion des différentes langues nous conduit à écarter également cette méthode.

4.2.5 Représentation phonétique des mots par la méthode Phonex

Contrairement aux algorithmes vus précédemment qui ont été conçus en premier lieu pour la langue anglaise puis adaptés aux spécificités de la langue française, l'algorithme Phonex (BROUARD 2004) fut lui directement conçu pour indexer des mots de la langue française.

L'algorithme Phonex procède en deux temps distincts pour le calcul de la représentation phonétique :

– Partie 1 : Représentation du mot par une suite finie de phonèmes

– Partie 2 : Encodage numérique de la suite de phonèmes en vue de l'indexation

Il se compose des 18 étapes énumérées en annexe C.1.3 page C-4.

Exemple 4.9

Les mots 'chair', 'cher', 'char', cri' et 'cru' ont tous un code Phonex différent constitué d'un attribut numérique unique compris entre 0 et 1.

Étape	***Code mot 1***	***Code mot 2***	***Code mot 3***
1-6	*chair*	*cher*	*char*
7-10	*chyr*	*chy*	*char*
11-12	*5yr*	*5y*	*5ar*
13	*5yr*	*5y*	*5ar*
14-16	*5yr*	*5y*	*5or*
17	*4,20,14*	*4,20*	*4,13,14*
18	$4.22^{-1}+20.22^{-2}+14.22^{-3}$	$4.22^{-1}+20.22^{-2}$	$4.22^{-1}+13.22^{-2}+14.22^{-3}$
18	0.225	0.224	0.211

Étape	*Code mot 4*	*Code mot 5*
1-6	*cri*	*cru*
7-10	*cri*	*cru*
11-12	*cri*	*cru*
13	*kri*	*kru*
14-16	*kri*	*kru*
17	*10,14,9*	*10,14,17*
18	$10.22^{-1}+14.22^{-2}+9.22^{-3}$	$10.22^{-1}+14.22^{-2}+17.22^{-3}$
18	0.485	0.486

Les mots « satisfait », « satifait », « prix » et « rpix » ont les codes Phonex suivants :

Étape	*Code mot 1*	*Code mot 2*
1-7	*satisfait*	*satifait*
7-13	*satisfyt*	*satifyt*
14-15	*sotisfyt*	*sotifyt*
16	*sotisfy*	*sotify*
17	*15,13,16,9,15,6,20*	*15,13,16,9,6,20*
18	$15.22^{-1}+13.22^{-2}+16.22^{-3}+9.22^{-4}+15.22^{-5}+6.22^{-6}+20.22^{-7}$	$15.22^{-1}+13.22^{-2}+16.22^{-3}+9.22^{-4}+6.22^{-5}+20.22^{-6}$
18	0.710	0.710

Étape	*Code mot 3*	*Code mot 4*
1-13	*prix*	*rpix*
14-15	*trix*	*rtix*
16	*tri*	*rti*
17	*16,14,9*	*14,16,9*
18	$16.22^{-1}+14.22^{-2}+9.22^{-3}$	$14.22^{-1}+16.22^{-2}+9.22^{-3}$
18	0.757	0.670

Nous pouvons observer sur ces exemples que les mots « satisfait » et « satifait », proches orthographiquement, ont la même représentation Phonex. Cela induira donc le rapprochement désiré entre ces mots. En revanche, les mots « prix » et « rpix » ont des codes Phonex bien distincts alors qu'ils sont proches également.

D'une manière générale, Phonex accorde plus d'importance au début du mot

qu'à la fin pour établir sa représentation. Les divergences en fin de mot comme pour le couple « satisfait / satifait » se traduiront donc par des représentations Phonex proches voir identiques alors que les divergences en début de mot comme pour le couple « prix / rpix » se traduiront par des représentations Phonex éloignées.

4.3 Relations de dissimilarité et similarité entre mots

Établir des distances, dissimilarités ou similarités entre les éléments de l'espace de représentation des mots permet de les mettre en relation et par exemple de proposer des corrections orthographiques.

Les distances, dissimilarités ou similarités s'appuient sur les attributs des différents modèles applicables sur cet espace. Ces modèles ont été décrits à la section 4.2.

4.3.1 Impact du modèle de représentation sur les distances, dissimilarités et similarités

Les modèles utilisés pour les représentations dans l'espace des mots sont séparés en deux groupes distincts :

1. Les modèles basés sur les caractères
2. Les modèles basés sur les phonèmes

Les premiers permettent le remplacement des mots non présents dans le lexique par un mot de la langue le plus proche en terme d'écriture.

Exemple 4.10

le style et la quamité des produits

Exemple 4.11

Magazin bizn organisé,vendeuse efficace en caisse.

Le mot « quamité » devrait ici être substitué par « qualité » avec ce type de représentation puisqu'il ne différencie que sur le caractère 'm'. De même, le mot « magazin » devrait être substitué par « magasin » et « bizn » par « bien ».

Les seconds modèles permettent le remplacement des mots non présents dans le lexique par un mot de la langue le plus proche en terme de phonétique.

Exemple 4.12

belle calite sur tout

Le mot « calite » devrait être ici substitué par « qualité » avec ce type de représentation.

4.3.2 Dissimilarité 0/1

Cette dissimilarité basique considère que seules les représentations strictement équivalentes sont semblables. Dans tous les autres cas, la dissimilarité entre les représentations est maximale.

Cette dissimilarité n'est que peu pertinente pour la majorité des représentations mais peut l'être dans le cas de représentations possédant une capacité de discrimination faible.

A la section 4.2.3, nous avons discuté de l'utilisation des représentations de la famille Soundex dans le cadre de la correction de langages textos. La dissimilarité 0/1 peut être employée afin de comparer de telles représentations (cf. tableau 4.2).

4.3.3 Dissimilarité entre représentations numériques

Lorsque les représentations sont simplement des valeurs numériques, nous pouvons avoir recours à une dissimilarité fondée sur la valeur absolue des différences.

C'est ainsi le cas dans le cadre de la représentation Phonex.

Exemple 4.13

Mot	*chair*	*cher*	*char*	*cri*	*cru*	*satisfait*	*satifait*	*prix*	*rpix*
Code Phonex	0.225	0.224	0.211	0.485	0.486	0.710	0.710	0.757	0.670

Les codes des mots 'chair', 'cher' et 'char' sont sensiblement liés : il n'y a en effet qu'une distance de 0.013 *entre les deux représentations des mots les plus éloignés. Le même constat peut être fait pour les mots « cri » et « cru » dont les représentations Phonex sont éloignées de* 0.001*. Les mots « satisfait » et « satifait »*

Mot	Code Soundex	Dissimilarité 0/1 sur le code Soundex	Code Soundex 2	Dissimilarité 0/1 sur le code Soundex 2
tjrs toujours	T768	0	TJR0	0
tt tout	T300	0	T000	0
slt salut	S430	0	SL00	0
ds dans	D800 D580	1	D000 DN00	1
cmd commande cmand	C530 C553	1 0	CM00 KMND CMN0	1 1

TABLE 4.2 – Dissimilarité 0/1 de codes Soundex et Soundex 2 de mots du français courant et de mots de textos français. Pour chaque ligne, les codes Soundex et Soundex 2 sont donnés dans la 2e et 4e colonne pour le mot de la 1re colonne. Les 3e et 4e colonne donnent les dissimilarités 0/1 entre les codes Soundex et Soundex 2 des mots de la 1re colonne.

> *ont des représentations Phonex identiques.*
>
> *Au contraire, les codes du groupe de mots « chair », « cher » et « char » et du groupe de mots « cri » et « cru » sont sensiblement différents : les représentations sont distantes au minimum de* 0.26*. Les mots « prix » et « rpix » ont des représentations Phonex distantes de* 0.087*.*

Nous voyons sur ces exemples que la distance appliquée à la représentation Phonex tend à rapprocher fortement les mots : elle ne fait ainsi presque aucune distinction entre « cri » et « cru » alors que les deux mots différent à la fin. Il en est de même pour les mots « satisfait » et « satifait ».

Le code Phonex accorde selon nous trop d'importance au début du mot alors que sa phonétique peut être autant erronée que le reste du mot (cas du couple « prix / rpix »). D'autre part, un décalage des sons suite par exemple à l'introduction d'un son « parasite » dans le début du mot, entraîne un changement important du code Phonex. Ce changement est lui aussi peu justifié selon nous. Comme nous le voyons ici, cette forte importance accordée au début du mot a un impact sur la dissimilarité établie (mots commençant par ch (« chair », « cher », « char »), mot commençant par

cr (« cri », « cru ») et mots commençant par sa (« satisfait », « satifait »)).

4.3.4 Dissimilarités d'édition

Lorsque les représentations sont des suites finies alphabétiques ou alphanumériques, nous pouvons avoir recours à des dissimilarités d'édition.

Notons $rep(m)$, la représentation par suite finie d'un mot m.

Définition 4.3 (*Dissimilarité d'édition*)

Une dissimilarité d'édition est une application de R^2 dans $\mathbb{R}^+$ avec R, l'ensemble des représentations.

$$\begin{array}{rccc} diss_{edit}: & R \times R & \rightarrow & \mathbb{R}^+ \\ & (rep(m_1), rep(m_2)) & \mapsto & diss_{edit}(rep(m_1), rep(m_2)) \end{array}$$

Nous présentons dans cette sous-section quelques dissimilarités d'édition.

Distance de Hamming

Définition 4.4 (*Distance de Hamming*)

*La distance de Hamming (*HAMMING *1950) entre la représentation d'un mot m_1 et la représentation d'un mot m_2 est donnée par le nombre de différences existantes entre les deux représentations. Elle vaut donc le nombre de substitutions nécessaires pour passer de la représentation du mot m_1 à celle de m_2. La substitution correspondant au remplacement d'un élément dans la représentation de m_1 par un nouvel élément pour se rapprocher de la représentation de m_2.*

Nous notons $d_H(rep(m_1), rep(m_2))$, la distance de Hamming entre les représentations des mots m_1 et m_2.

Exemple 4.14

Prenons en exemple, la distance de Hamming entre les représentations par suites finies de caractères des mots « char » et « cher ».

Étape	*Représentation atteinte*	*Distance*
1	*(c, h, **a**, r)*	*0*
2	*(c, h, **e**, r)*	*1*

Le a du mot « char » est substitué par le e du mot « cher » donnant ainsi une distance de Hamming de 1 entre les deux mots.

Distance de Levenshtein

Définition 4.5 (*Distance de Levenshtein*)

*La distance de Levenshtein (*LEVENSHTEIN *1966) entre la représentation d'un mot m_1 et la représentation d'un mot m_2 est donnée par le nombre minimal de changements « primaires » pour passer de la représentation du mot m_1 à celle du mot m_2. Ces changements « primaires » sont au nombre de trois :*

- *L'insertion : on ajoute un élément supplémentaire dans la représentation de m_1 pour se rapprocher de la représentation de m_2 ;*
- *La suppression : on supprime un élément dans la représentation de m_1 pour se rapprocher de la représentation de m_2 ;*
- *La substitution : on remplace un élément dans la représentation de m_1 par un nouvel élément pour se rapprocher de la représentation de m_2.*

Nous notons $d_L(rep(m_1), rep(m_2))$, la distance de Levenshtein entre les représentations des mots m_1 et m_2.

Comme la distance de Levenshtein inclut la substitution, nous avons la propriété remarquable que :

$$\forall (rep(m_1), rep(m_2)) \in R^2, d_L(rep(m_1), rep(m_2)) \leq d_H(rep(m_1), rep(m_2))$$

Une variante de cette distance propose de retirer la substitution des changements « primaires ». Toute substitution peut en effet se réaliser par la succession des deux autres actions mais cela engendre la perte de la satisfaction de l'inégalité précédente.

Exemple 4.15

Étudions en exemple la distance de Levenshtein entre les représentations par

suites finies de caractères des mots « char » et « cher ».

Étape	***Représentation atteinte***	***Distance***
1	*(c, h,* ***a****, r)*	*0*
2	*(c, h,* ***e****, r)*	*1*

Le a du mot « char » est substitué par le e du mot « cher » donnant ainsi une distance de Levenshtein de 1 entre les deux mots identique à celle de Hamming. Si l'on supprime la substitution des changements primaires, la variante de la distance de Levenshtein donne :

Étape	***Représentation atteinte***	***Distance***
1	*(c, h,* ***a****, r)*	*0*
2	*(c, h, r)*	*1*
3	*(c, h,* ***e****, r)*	*2*

Le a du mot « char » est supprimé, puis le e du mot « cher » est ajouté donnant ainsi une distance de 2 entre les deux mots.

La distance entre les deux représentations des deux mots est donc doublée selon que l'on permet ou non la substitution comme un changement primaire.

Cette variante correspond en fait à un cas particulier où chaque changement « primaire » possède un coût pouvant être différent de 1. Ce coût peut être fonction du type de changement (insertion, suppression, substitution) mais également fonction des caractères affectés dans la suite par ce changement. Il est ainsi possible de considérer la substitution d'un caractère avec diacritique pour son homologue sans diacritique comme un changement moins coûteux que les autres.

D'une manière générale, l'inégalité précédente est satisfaite si chacun des trois changements « primaires » de Levenshtein possède un coût inférieur ou égal à 1.

D'un point de vue calculatoire, Le calcul de la distance de Levenshtein se résout par le problème du calcul d'une matrice $L^{rep(m_1),rep(m_2)}$ de taille $(|rep(m_1)|+1) \times (|rep(m_2)|+1)$.

Ce calcul peut être coûteux s'il est entrepris pour l'ensemble des couples $(rep(m_1), rep(m_2)) \in R^2$. Cependant, la matrice possède la particularité intéressante suivante :

$$\forall (i,j) \in [\![1, |rep(m_1)|-1]\!] \times [\![1, |rep(m_2)|-1]\!], L^{rep(m_1),rep(m_2)}_{\{1,i+1\}\{1,j+1\}} = L^{(rep(m_1))_{[\![1,i]\!]},rep(m_2)_{[\![1,j]\!]}}$$

Il est ainsi possible d'avoir recours à la programmation dynamique afin de ne calculer qu'une seule fois chaque sous-matrice distincte et limiter ainsi le nombre de calcul.

Distance de Damerau-Levenshtein

Définition 4.6 (*Distance de Damerau-Levenshtein*)

*La distance de Damerau-Levenshtein (*DAMERAU *1964) entre la représentation d'un mot m_1 et la représentation d'un mot m_2 est donnée par le nombre minimal de changements « primaires » pour passer de la représentation du mot m_1 à celle du mot m_2. Ces changements « primaires » sont au nombre de quatre dont les trois de la distance de Levenshtein :*

- *L'insertion*
- *La suppression*
- *La substitution*

auxquels s'ajoute la transposition, inversion de deux caractères voisins.

Nous notons $d_{DL}(rep(m_1), rep(m_2))$, la distance de Damerau-Levenshtein entre les représentations des mots m_1 et m_2.

Comme la distance de Damerau-Levenshtein inclut la substitution, l'insertion et la suppression, nous avons la propriété remarquable que :

$$\forall (rep(m_1), rep(m_2)) \in R^2,$$
$$d_{DL}(rep(m_1), rep(m_2)) \leq d_L(rep(m_1), rep(m_2)) \leq d_H(rep(m_1), rep(m_2))$$

L'inégalité précédente reste satisfaite pour les variantes de la distance de Damerau-Levenshtein incluant des coûts pour chaque changement tant que chacun des quatre changements primaires possède un coût inférieur ou égal à 1.

Exemple 4.16

Prenons en exemple, la distance de Damerau-Levenshtein entre les représentations par suites finies de caractères des mots « cher » et « chre ».

Étape	***Représentation atteinte***	***Distance***
1	*(c, h,* ***r****,* ***e****)*	*0*
2	*(c, h,* ***e****,* ***r****)*	*1*

Les caractères r et e du mot « chre » sont transposés pour constituer le mot « cher » donnant ainsi une distance de Damerau-Levenshtein de 1 entre les deux mots. Si l'on supprime la substitution des changements primaires, la variante de la distance de Damerau-Levenshtein donne le même résultat avec une distance inchangée de 1.

Calculons maintenant en comparaison la distance de Levenshtein entre les deux mêmes représentations.

Étape	***Représentation atteinte***	***Distance***
1	*(c, h, r, e)*	*0*
2	*(c, h, **e**, r, **e**)*	*1*
3	*(c, h, e, r)*	*2*

La distance de Levenshtein est donc de 2 donnée par l'insertion du caractère 'e' en 3e position suivi par la suppression de ce même caractère en 5e position. Encore une fois, la variante de Levenshtein supprimant la substitution ne change rien à cette distance.

Similarité de Jaro

Définition 4.7 (*Similarité de Jaro*)

*La similarité de Jaro (*JARO *1989) entre la représentation d'un mot m_1 et la représentation d'un mot m_2 est définie par :*

$$sim_{Jaro} = \frac{1}{3}\left(\frac{m}{|rep(m_1)|} + \frac{m}{|rep(m_2)|} + \frac{m-t}{m}\right)$$

où :

1. *$|rep(m_i)|$ est la longueur de la représentation de m_i ;*
2. *m est le nombre de caractères correspondants ;*
3. *t est le nombre de transpositions.*

Deux caractères identiques de $rep(m_1)$ et de $rep(m_2)$ sont considérés comme correspondants si leur éloignement (i.e. la différence entre leurs positions dans leurs chaînes respectives) ne dépasse pas :

$$\left\lfloor \frac{\max(|rep(m_1)|, |rep(m_2)|)}{2} \right\rfloor - 1$$

Le nombre de transpositions est obtenu en comparant le i-ème caractère correspondant de $rep(m_1)$ avec le i-ème caractère correspondant de $rep(m_2)$. Le nombre de fois où ces caractères sont différents, divisé par deux, donne le nombre de transpositions.

Exemple 4.17

Calculons à présent la similarité de Jaro entre les représentations par suite de caractères des mots « char » et « cher ».

Les caractères correspondants sont au nombre de trois (c, h, r) pour les deux représentations. Comme tous les caractères correspondants dans les deux chaînes sont identiques, il n'y a pas de transpositions. Par ailleurs, la longueur des deux représentations est de 4. Cela nous donne donc une similarité de Jaro de

$$sim_{Jaro} = \frac{1}{3}\left(\frac{3}{4} + \frac{3}{4} + \frac{3-0}{3}\right) = 0.833$$

Similarité de Jaro-Winkler

Définition 4.8 (*Similarité de Jaro-Winkler*)

*La similarité de Jaro-Winkler (*WINKLER *1999) entre la représentation d'un mot m_1 et la représentation d'un mot m_2 se base sur la similarité de Jaro en privilégiant davantage (i.e., en augmentant la similarité) les représentations ayant un préfixe commun. Elle fut proposée par Winkler comme une extension de la similarité de Jaro. Soient l, la longueur du préfixe commun aux deux représentations et p, un coefficient de privilégiation. La similarité de Jaro-Winkler est alors donné par :*

$$sim_{Jaro-Winkler} = sim_{Jaro} + (l \times p \times (1 - sim_{Jaro}))$$

Winkler propose par ailleurs de prendre $l \leq 4$ et $p = 0.1$.

Exemple 4.18

La similarité de Jaro-Winkler entre les représentations par suite de caractères des

mots « char » et « cher » obtenue en prenant $l = 2$ *et* $p = 0.1$ *vaut :*

$$\begin{aligned} sim_{Jaro-Winker} &= sim_{Jaro} + (l \times p \times (1 - sim_{Jaro})) \\ &= 0.833 + (2 \times 0.1 \times (1 - 0.833)) \\ &= 0.8664 \end{aligned}$$

4.4 Conclusion

Dans ce chapitre, nous avons présenté une formalisation du processus de correction orthographique par la résolution de similarités ou dissimilarités existantes entre des représentations de mots.

Nous avons considéré trois types de représentations : la représentation par suite de caractères, les représentations consonantiques de la famille des Soundex et la représentation phonétique Phonex.

Nous avons également étudié plusieurs relations de similarités et dissimilarités. L'utilité de chacune est fonction de la nature de la représentation utilisée et surtout de ses capacités de distinction de différents mots proches d'un point de vue orthographique.

Chapitre 5

Représentations vectorielles et dissimilarités colorimétriques

De même que les textes, les couleurs peuvent être représentés par des vecteurs. Il est alors envisageable d'utiliser des relations de dissimilarités vectorielles entre ces représentations colorimétriques telles que celles que nous avons présentées au chapitre 2.

Lorsque l'on recherche des couleurs « proches » d'une certaine couleur dans une base de couleurs, la relation de dissimilarité entre couleurs doit refléter la proximité perceptible entre les couleurs par un humain.

Contrairement aux textes, les représentations vectorielles colorimétriques ne disposent pas nécessairement de beaucoup d'attributs. Nous étudions des systèmes de représentations des couleurs utilisant uniquement trois dimensions. L'espace vectoriel obtenu est alors facilement modélisable par coupes. La distance euclidienne est sans doute dans ce contexte la dissimilarité la plus intuitive à utiliser et permet ainsi une validation aisée par des humains et à grande échelle du modèle obtenu.

Le problème qui se pose alors est d'identifier un espace vectoriel colorimétrique de trois dimensions sur lequel la distance euclidienne est en tout point représentative de la distance colorimétrique perçue.

Nous présentons, dans ce chapitre, des méthodes standard de représentations

vectorielles numériques pour les couleurs et nous attardons sur l'adéquation de la distance euclidienne avec la perception des couleurs.

5.1 Perception colorimétrique

Avant de nous intéresser aux représentations vectorielles colorimétriques, nous présentons dans cette section la perception des couleurs. Ces notions sont importantes dans le sens où la représentation vectorielle recherchée doit refléter cette perception dans sa distance euclidienne.

5.1.1 Longueur d'ondes et spectre de la lumière visible

La perception de la couleur est rendue possible par les différentes longueurs d'ondes de la lumière visible (onde électromagnétique).

Le spectre de la lumière visible désigne l'ensemble des longueurs d'ondes que prend la lumière visible. Les longueurs d'ondes du spectre sont comprises entre 380 et 740 nanomètres et la couleur perçue est caractérisée par la valeur de la longueur d'ondes et l'intensité de la lumière. Lorsque la longueur d'ondes est supérieure à 740 nanomètres ou inférieure à 380 nanomètres, il n'y a ni couleur ni lumière visible pour les humains.

Par la suite, la lumière visible est appelée simplement lumière.

5.1.2 Trichromatisme et tetrachromatisme

Il y a deux sortes de perception des couleurs dénotées par trichromatisme et tetrachromatisme. Ces termes sont liés à la manière de percevoir les couleurs. Trichromatisme signifie que trois capteurs différents sont utilisés dans leur perception alors qu'il y en a quatre en tetrachromatisme.

En conséquence directe, les tetrachromates, qui utilisent quatre capteurs différents, peuvent percevoir plus de couleurs différentes que les trichromates, qui uti-

lisent trois capteurs différents, s'ils sont utilisés dans la même plage de lumière perceptible.

Le plus souvent, le tetrachromatisme est corrélé à la capacité de percevoir la lumière non visible. Dans ce cas, trois capteurs sont utilisés pour percevoir la lumière visible et le dernier pour percevoir la lumière non visible comme les ultraviolets.

Les humains sont trichromates. Nous étudions plus précisément dans la sous-section suivante la perception trichromatique utilisée par les humains.

5.1.3 Perception trichromatique humaine

La couleur est perçue par les humains au travers de l'intensité et de la longueur d'onde de la lumière reçue par les yeux et perçue par un système trichromatique de cônes (YOUNG 1802).

Les couleurs perceptibles par les humains sont toutes situées approximativement entre 380 et 700 nanomètres dans le spectre de la lumière visible. Les études ont montré qu'environ 150 longueurs d'ondes sont perceptibles par les humains (WEIL-BARAIS et DUBOIS 1994).

Plus précisément, la vision humaine des couleurs est basée sur un tri-stimuli rendu possible par trois types de cônes qui sont chacun photosensibles à une partie limitée des longueurs d'ondes de la lumière (WRIGHT 1929) (GUILD 1932). Toutes les couleurs visibles peuvent être exprimées en tant que composition de ces trois rangs de longueur d'ondes. Chacun de ces trois types de cônes a alors une réponse pour une excitation lumineuse en accord avec l'intensité du signal pour chacune des trois composantes.

Des modèles et des espaces colorimétriques ont été développés à partir des théories de la perception. Nous les décrivons dans la prochaine section et les mettons en rapport aux modèles et espaces utilisés en linguistique.

5.2 Espaces et modèles colorimétriques

Un ensemble de couleurs peut être représenté dans un espace communément appelé espace colorimétrique. Ces espaces, munis d'une norme, permettent d'établir des dissimilarités entre les couleurs. Cette norme est définie au travers de modèles colorimétriques basés soit sur la linguistique, soit sur les théories de la perception discutées à la section 5.1. Nous montrons les avantages des secondes pour notre traitement.

5.2.1 Espaces et modèles colorimétriques à partir de relations linguistiques

Un modèle colorimétrique basé sur des relations linguistiques s'appuie sur la similarité entre couleurs définie dans une ou plusieurs ressources linguistiques. Cette similarité est déterminée selon la perception globale des entités physiques désignées par les termes. Par exemple, la couleur lave est similaire à la couleur rouge en raison d'une perception similaire de la couleur de la lave et du rouge. Cette similarité peut être obtenue depuis une ressource linguistique généraliste telle que *Wordnet* en utilisant les relations linguistiques comme la synonymie et/ou l'hyponymie comme proximités entre les noms de couleurs.

Néanmoins, si le modèle est relativement simple à construire depuis une ressource linguistique parfaite, cette solution est limitée en trois points.

Premièrement, des ressources linguistiques de qualité suffisante ne sont pas librement disponibles dans toutes les langues. En anglais, le *Wordnet* de Princeton (MILLER 1995) a atteint une maturité suffisante pour traiter les couleurs mais pour le français, par exemple, il n'y a aucune ressource de qualité équivalente.

Deuxièmement, les ressources linguistiques comme *Wordnet* sont difficiles à mettre à jour de manière automatique. Par exemple, si nous cherchons le *synset* du champ chromatique rouge, nous trouvons neuf *synsets* hyponymiques regroupant des couleurs dérivées du champ chromatique. Malheureusement, même si l'un d'eux contient la couleur vermillon, aucun ne contient directement ou dans ses propres *synsets* hy-

ponymiques la couleur lave mentionnée auparavant. Devrait-elle être ajoutée dans un *synset* existant (comme synonyme de vermillon par exemple) ou dans un nouveau ? Si un nouveau *synset* est créé, comment doit-il être intégré par rapport aux autres *synsets* ? Ces questions n'ont pas de réponse triviale même dans un traitement manuel. De ce fait, l'évolutivité requise du système ne peut pas être garantie avec cette méthode.

Troisièmement, le modèle colorimétique utilisé ici n'est pas un modèle vectoriel. Il n'est donc pas possible d'utiliser une dissimilarité classique sur les espaces vectoriels comme la distance euclidienne. Ce problème limite la capacité de validation du système et de la dissimilarité obtenu.

Ces trois raisons nous font préférer les espaces et modèles vectoriels pour nos travaux.

5.2.2 Espaces et modèles vectoriels colorimétriques à partir d'informations chromatiques

Grâce aux informations chromatiques sur les couleurs issus des théories perceptives présentées à la section 5.1, des modèles vectoriels colorimétriques ont été mis au point.

Le plus ancien modèle a été fondé sur la théorie de T. Young de la vision trichromatique (YOUNG 1802). Celui-ci, appelé Rouge Vert Bleu (RVB), est le plus connu de nos jours. Il est ainsi largement utilisé pour représenter des couleurs dans le domaine de l'informatique. Les couleurs sont décrites dans ce modèle comme un mélange additif des trois couleurs primaires. L'addition complète des trois couleurs donne un blanc pur tandis que le noir pur est donné par la privation des trois couleurs. Les propriétés rouge, verte et bleu sont usuellement exprimées entre 0 pour une privation de la couleur et 1 pour une présence complète de celle-ci.

De nombreux espaces colorimétriques ont été définis à partir du modèle RVB. Nous nous intéressons seulement aux espaces CIE RVB et RVB standard (sRVB).

L'espace CIE RVB a été spécifié depuis les observations de Wright et Guild (WRIGHT 1929) (GUILD 1932) par la Commission Internationale de l'Eclairage (CIE) en 1931.

Cet espace colorimétrique a été conçu dans l'objectif de représenter l'ensemble des couleurs visibles avec une distance entre elles représentative de la perception trichromatique RVB. Il n'est en général pas utilisé avec le modèle RVB qui a permis sa construction mais avec le modèle CIE 1931 XYZ défini par la CIE la même année (SMITH et GUILD 1931). La principale raison du changement de modèle vient du fait que contrairement à RVB, le modèle XYZ peut exprimer l'ensemble des couleurs de l'espace colorimétrique avec des valeurs d'attributs positives. Il est usuel de normaliser deux de ces attributs. Le modèle xyY est ainsi défini par $x = X/(X+Y+Z)$ et $y = Y/(X+Y+Z)$. Le modèle XYZ fut amélioré plus tard par des modèles dérivés tels que Hunter Lab en 1948 (HUNTER 1948) ou les modèles CIE Lab et CIE Luv tous deux en 1976.

L'espace RVB standard (sRVB) a été spécifié plus tard par Hewlett-Packard et Microsoft en 1996 (STOKES et al. 1996). Celui-ci a été spécialement défini dans le but de représenter les couleurs sur le Web. De plus, il est usuellement utilisé pour afficher les couleurs sur un écran. Il correspond à un sous-espace de CIE RVB. Il en résulte qu'il est possible d'utiliser les modèles CIE 1931 XYZ et CIE Lab sur cet espace colorimétrique. Les figures 5.1 et 5.2 montrent des coupes de l'espace colorimétrique sRVB avec les modèles xyY et CIE Lab.

Le modèle CIE Lab est spécialement connu en tant que modèle transcrivant de manière uniforme la perception des couleurs par trichromatisme comme la perception humaine. Nous pouvions donc attendre de ce modèle de définir une distance euclidienne entre les couleurs conforme en tout point de l'espace à la perception des couleurs. Même si cela est avéré en de nombreux points, l'étude du modèle montre des problèmes pour certaines régions de l'espace : les zones des couleurs blanches, noires et grises. Le blanc est représenté par un point seul et le noir sur toute la zone sud pour les plus petites valeurs de L. Les gris sont situés dans le centre de l'espace alors que les autres couleurs sont distribuées par ailleurs. Ces observations suggèrent un déséquilibre entre l'espace occupé par le blanc, le noir, les gris et les autres couleurs, perturbant par la même l'homogénéité colorimétrique modélisée par une distance euclidienne. De plus, le modèle CIE Lab n'est pas parmi les modèles les plus intuitifs, contrairement à ceux que nous décrivons par la suite.

Dans le modèle Cyan Magenta Jaune (CMJ), les couleurs sont décrites comme

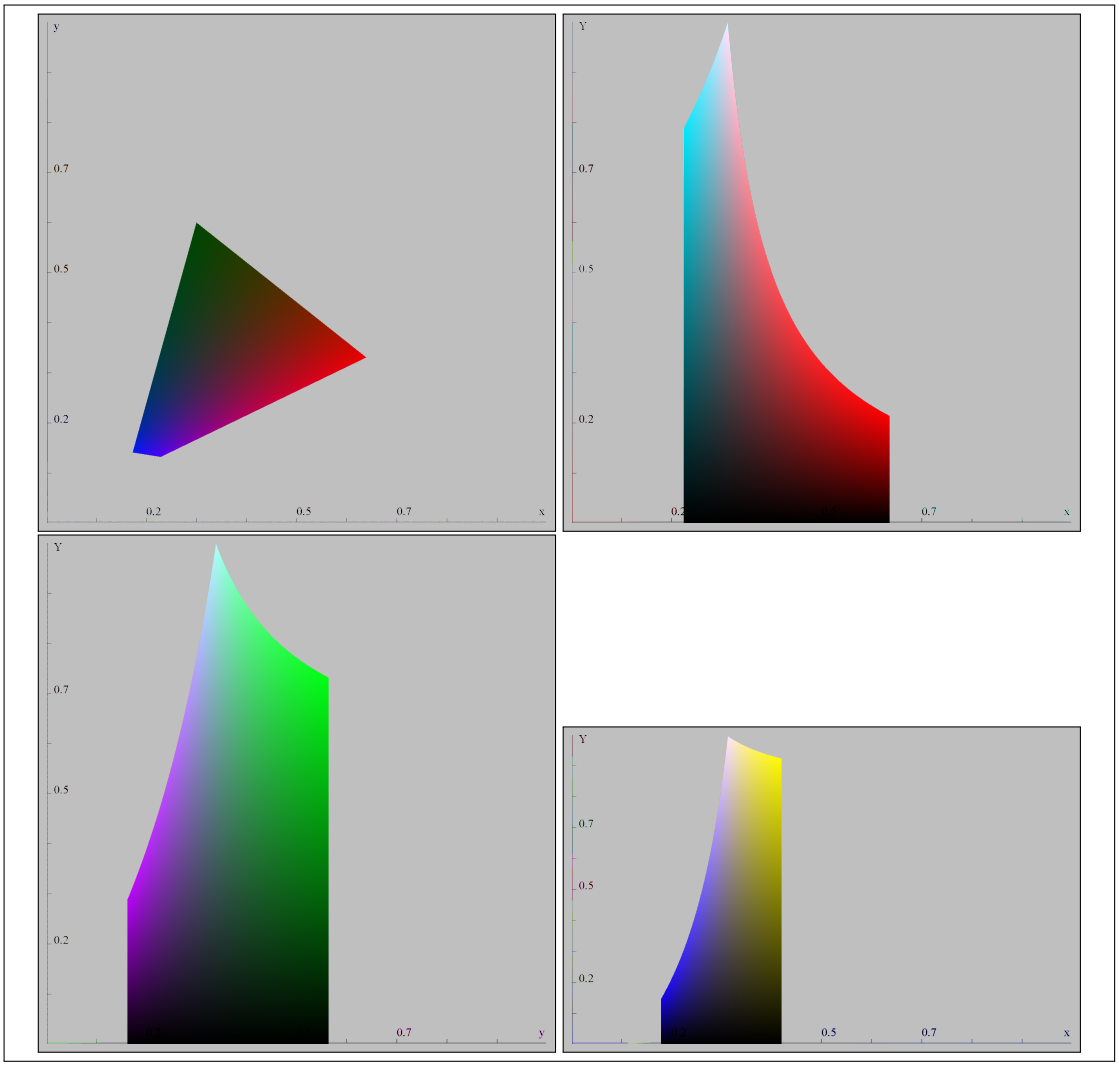


FIGURE 5.1 – Coupes de l'espace colorimétrique sRVB avec le modèle xyY. Les 4 figures présentent de gauche à droite et de haut en bas, une coupe horizontale puis des coupes verticales de l'espace colorimétrique sRVB avec le modèle xyY.

un mélange des trois couleurs primaires soustractives. La complète soustraction des trois couleurs au blanc donne une couleur tendant sur le noir alors que le blanc pur est obtenu naturellement en ne soustrayant aucune des trois primaires au blanc. Les propriétés cyan, magenta et jaune sont usuellement exprimées entre 0 pour une absence de privation et 1 pour une privation complète de la primaire au blanc. Ce modèle correspond aux théories de perception trichromatique de la lumière blanche réfléchie par un objet. Néanmoins, pour notre travail, la réflection de la lumière ne

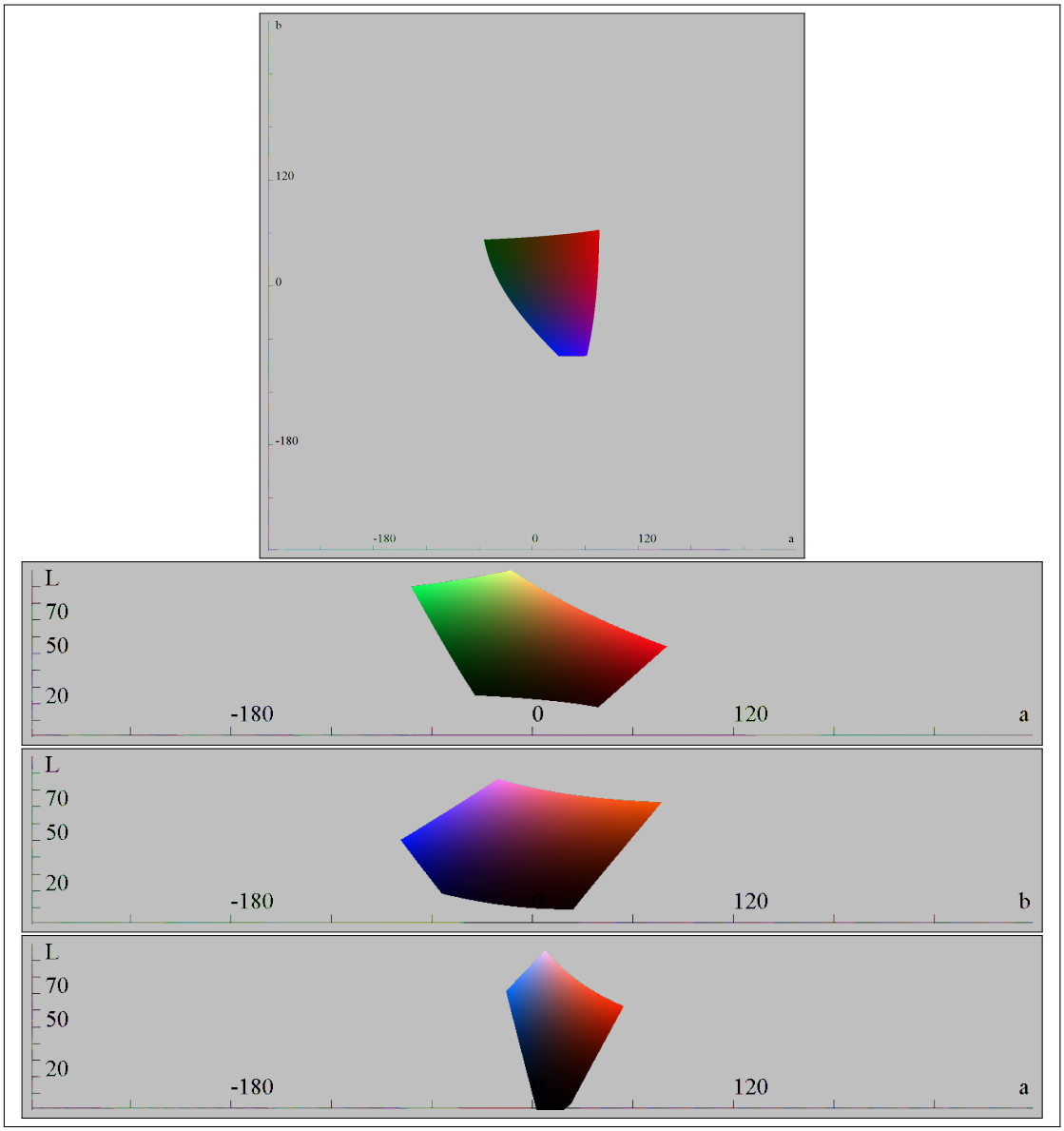


FIGURE 5.2 – Coupes de l'espace colorimétrique sRVB avec le modèle CIE Lab. Les 4 figures présentent de haut en bas, une coupe horizontale puis des coupes verticales de l'espace colorimétrique sRVB avec le modèle CIE Lab.

semble pas avoir une grande importance puisque seule la perception de la couleur l'est.

D'une autre manière, certains modèles n'utilisent pas directement les théories de la perception mais d'autres propriétés perceptuelles dérivées de ces dernières. Le

Teinte Saturation Luminosité (TSL) et le Teinte Saturation Valeur (TSV) sont largement utilisés dans ce sens. La Valeur du modèle TSV correspondant à la brillance de la couleur, nous préférerons parler de Brillance du modèle TSV. Les deux modèles sont dérivés du modèle RVB. Ils utilisent les propriétés Teinte et Saturation dans leurs attributs pour décrire les couleurs. La propriété Teinte représente la teinte de la couleur et plus précisément le degré de la teinte sur le cercle chromatique. Il est exprimé en degrés entre 0 et 360. La propriété Saturation représente l'intensité de la teinte de la couleur par rapport à sa Luminosité ou sa Brillance. La Saturation est exprimée entre 0 pour les couleurs non saturées et 1 pour les couleurs saturées. Néanmoins, les valeurs de la Saturation sont différentes pour une même couleur dans les deux modèles. Cela est dû à la dernière propriété utilisée par les modèles. Celle-ci est différente : le modèle TSL utilise la Luminosité alors que le modèle TSV utilise la Brillance. La propriété Luminosité transcrit la quantité de lumière dans la couleur (Luminance est aussi utilisée pour cette propriété). La Luminosité est exprimée entre 0 pour les couleurs sombres et 1 pour les couleurs lumineuses. La propriété Brillance transcrit la brillance de la couleur. La propriété Brillance est exprimée entre 0 pour les couleurs sombres et 1 pour les couleurs brillantes. Ces deux modèles forment un cylindre. Les figures 5.3 et 5.4 donnent des coupes des modèles TSL et TSV dans l'espace colorimétrique sRVB. Plus de détails sur ces modèles et comparaisons entre eux sont discutés dans les travaux de Joblove (JOBLOVE et GREENBERG 1978).

Ces modèles sont imparfaits pour la représentation du blanc, du noir et des gris. Comme nous le voyons dans la coupe verticale, blanc et noir sont représentées par des zones rectangulaires aux pôles nord et sud du cylindre du modèle TSL et par une zone rectangulaire dans le pôle sud pour le noir et par un seul point dans le centre du pôle nord pour le blanc dans TSV. De plus, comme nous le voyons clairement sur la coupe horizontale, les gris sont présents uniquement dans le centre du cylindre alors que les autres couleurs sont dans le reste du cylindre. Ces observations montrent un déséquilibre entre l'espace occupé par le blanc, les gris, le noir et les autres couleurs.

Au lieu de la propriété Saturation, il est possible d'utiliser la propriété de Chrominance (Chroma) dans les deux modèles. La chrominance est l'intensité de la teinte de la couleur par rapport aux couleurs non chromatiques (typiquement, le blanc et le noir) (JOBLOVE et GREENBERG 1978). Même si elle varie entre 0 et 1 comme la satu-

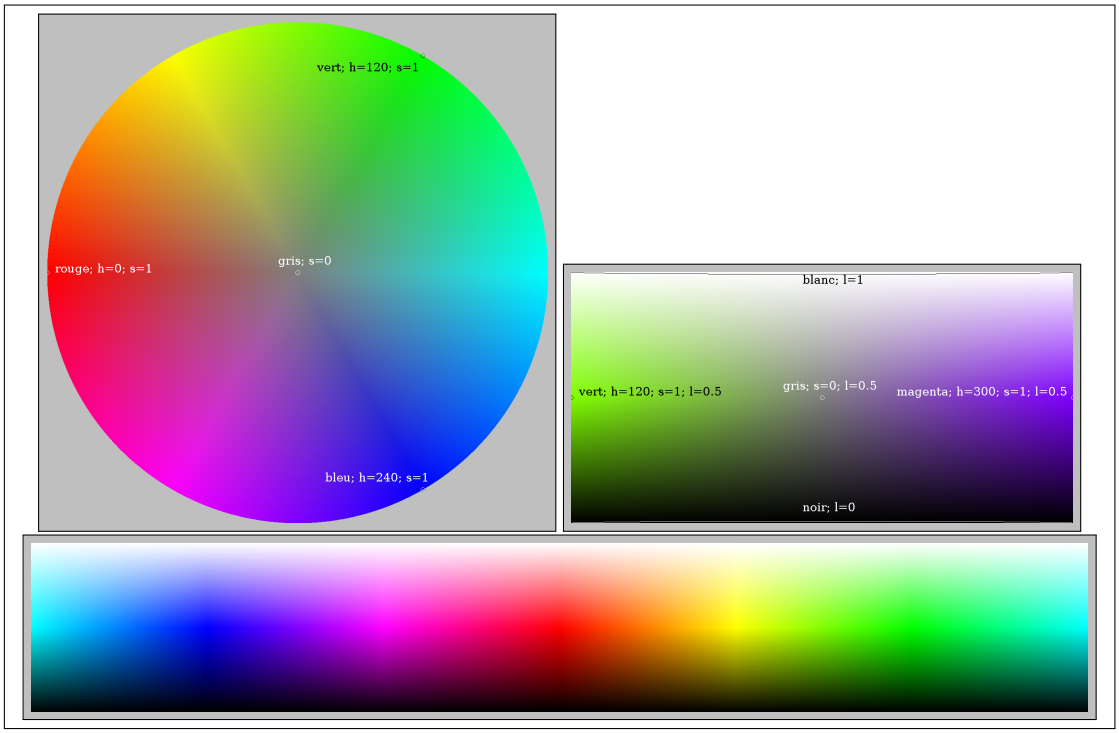


FIGURE 5.3 – Coupes de l'espace colorimétrique sRVB avec le modèle TSL. Les 3 figures présentent de gauche à droite et de haut en bas, des coupes horizontale, verticale et circulaire de l'espace colorimétrique sRVB avec le modèle TSL.

ration, la principale différence avec la saturation réside dans la valeur maximale que peut prendre la propriété pour une Luminosité ou Brillance donnée. La Saturation atteint 1 dans tous les cas puisque la mesure est effectuée par rapport à la Luminosité ou Brillance considérée mais la Chroma atteint seulement 1 pour une Luminosité moyenne et une Brillance maximale et décroit dans les autres cas pour être forcée à 0 pour le blanc et le noir. Soit une couleur défini par le triplet (r, v, b) dans le modèle RVB, Nous pouvons définir la Chroma :

$$c = max(r, v, b) - min(r, v, b)$$

Les figures 5.5 et 5.6 donnent des coupes des modèles TCL et TCV.

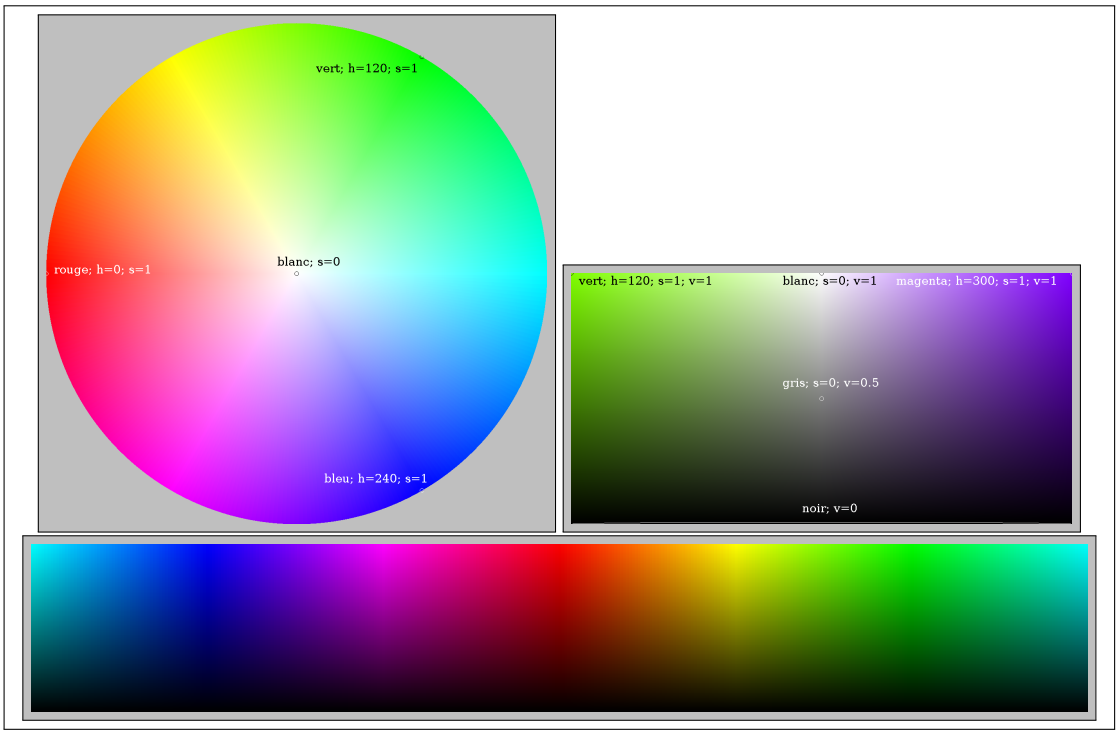


FIGURE 5.4 – Coupes de l'espace colorimétrique sRVB avec le modèle TSV. Les 3 figures présentent de gauche à droite et de haut en bas, des coupes horizontale, verticale et circulaire de l'espace colorimétrique sRVB avec le modèle TSV.

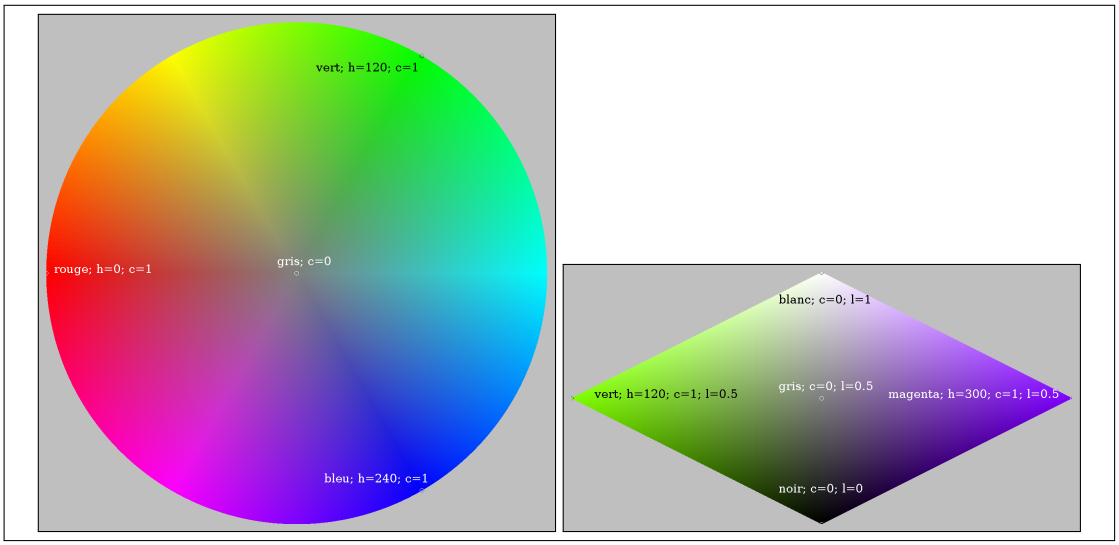


FIGURE 5.5 – Coupes de l'espace colorimétrique sRVB avec le modèle TCL. Les 2 figures présentent de gauche à droite, des coupes horizontale et verticale de l'espace colorimétrique sRVB avec le modèle TCL.

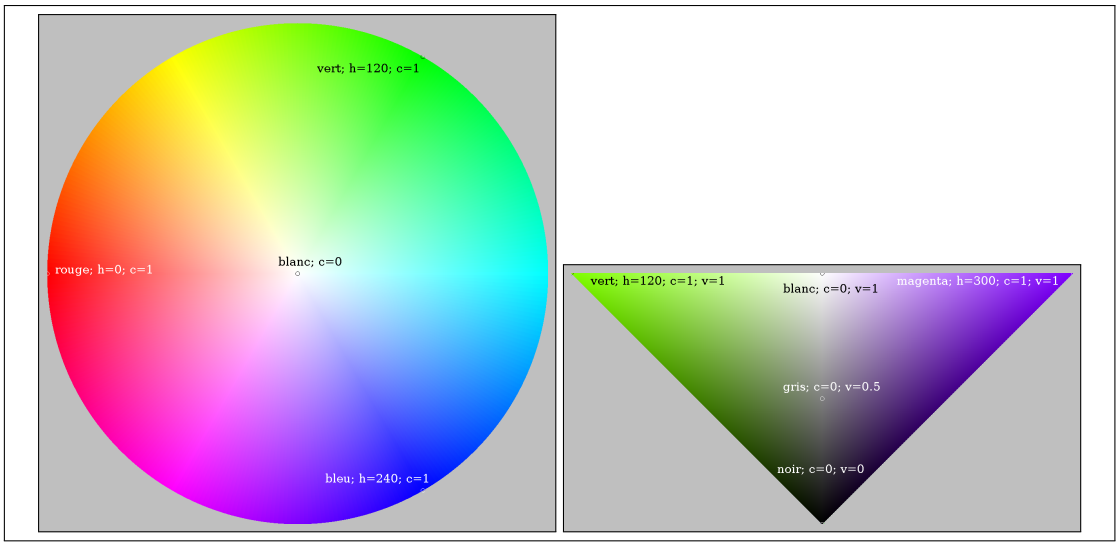


FIGURE 5.6 – Coupes de l'espace colorimétrique sRVB avec le modèle TCV. Les 2 figures présentent de gauche à droite, des coupes horizontale et verticale de l'espace colorimétrique sRVB avec le modèle TCV.

Deuxième partie

Apprentissage artificiel

Chapitre 6

Apprentissage artificiel : classification et regroupements

Dans cette partie, nous présentons plusieurs notions relatives à l'apprentissage artificiel et utilisées dans la suite. Ces définitions reprennent celles données dans (CORNUÉJOLS et MICLET 2010).

6.1 Classification et regroupement

6.1.1 Classe, recouvrement, partition et fonction de classement

Les tâches de classification et regroupement sont au cœur de nos problématiques sur les textes. Nous commençons par rappeler la définition de la notion de classe, notion centrale pour ces tâches.

Définition 6.1 (*Classe*)

Une classe P est un sous-ensemble non vide d'un ensemble, noté X, dont les éléments ont été choisis en fonction d'une régularité. Nous avons les propriétés :

partie avec ensemble vide exclu : $P \in 2^X \setminus \{\emptyset\}$

A partir des classes, nous pouvons effectuer des recouvrements.

Définition 6.2 (*Recouvrement*)

Un recouvrement d'un ensemble X est un ensemble de classes, donc un ensemble de parties $\rho = \{P_1, P_2, ..., P_n\}$ de l'ensemble X tel que :

$$\textit{recouvrement} : \bigcup_{i=1}^{n} P_i = X$$

ρ doit vérifier :

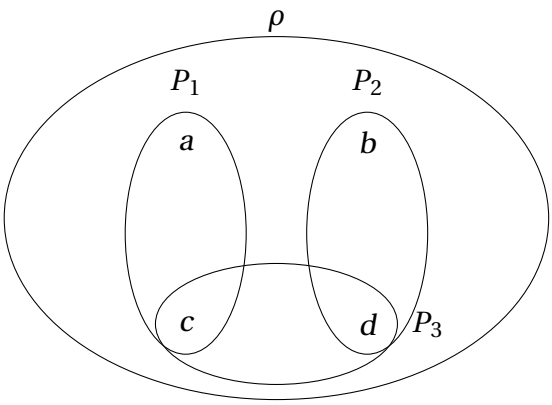
$$\rho \subseteq 2^X \setminus \{\emptyset\}$$

Le nombre de classes dans un recouvrement, noté K, possède nécessairement la propriété suivante :

$$K \leq |2^X \setminus \{\emptyset\}|$$

Exemple 6.1

La figure suivante donne un exemple de recouvrement ρ constitué des 3 classes P_1, P_2 et P_3.



De manière restreinte par rapport au recouvrement, nous pouvons également établir une partition à partir des classes.

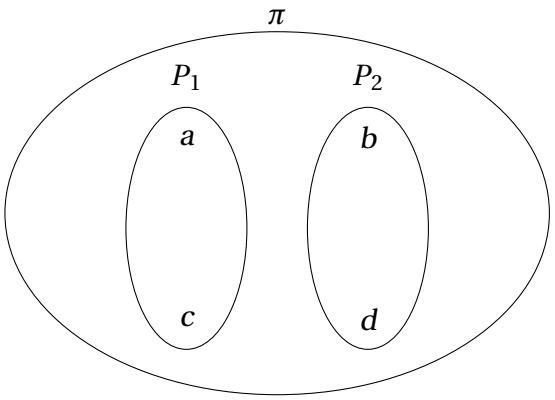
Définition 6.3 (*Partition*)

Une partition d'un ensemble X est un recouvrement $\pi = \{P_1, P_2, ..., P_n\}$ dont les classes sont deux à deux disjointes :

$$\text{classes disjointes : } \forall (P_i, P_j) \in \pi^2, P_i \neq P_j \Rightarrow P_i \cap P_j = \emptyset$$

Exemple 6.2

La figure suivante donne un exemple de partition π constituée des 2 classes P_1 et P_2.



La classe P_3, définie au préalable, ne peut être incluse ou ajoutée à la partition car elle ne respecte pas la propriété de disjonction avec les deux autres classes.

Nous pouvons à présent introduire la notion de fonction de classement.

Définition 6.4 (*Fonction de classement, application de décision*)

Une fonction de classement ou application de décision, notée h, appartenant à un ensemble d'hypothèses H, est une application définie d'un ensemble d'entrées X dans un ensemble de sorties Y.

$$h : X \rightarrow Y$$

et ayant pour objectif de classer les instances de l'ensemble X dans une ou plusieurs classes définies dans l'ensemble Y. Plus formellement, la fonction h est

une application établissant un recouvrement ou une partition de l'ensemble X.

6.1.2 Regroupement

Introduisons la notion de regroupement.

Définition 6.5 (*Regroupement*)

Le regroupement est la tâche visant à établir une fonction de classement d'un ensemble d'entrée sans connaissances a priori sur les classes. De manière plus formelle, une tâche de regroupements consiste en l'élaboration d'une fonction de classement de l'ensemble d'entrée dans l'ensemble des parties non vides de l'ensemble des parties non vides de l'ensemble d'entrée :

$$h : X \rightarrow 2^{2^X \setminus \{\emptyset\}} \setminus \{\emptyset\}$$

Lorsque l'ensemble d'entrée X devient très grand, comme cela est le cas lorsqu'on traite des textes, il est impossible d'explorer $2^{2^X \setminus \{\emptyset\}} \setminus \{\emptyset\}$. Néanmoins, le nombre de classes souhaités dans le regroupement est souvent limité à un nombre raisonnable. Dans ce contexte, l'ensemble de sortie est l'ensemble des parties non vides de l'ensemble W constitué de K classes w_i à identifier sur X. Le profil de h devient :

$$h : X \rightarrow 2^W \setminus \{\emptyset\}$$

avec :

$$W = \{w_1, ..., w_K\} \tag{6.1}$$

Il en résulte la propriété suivante :

$$W \subseteq 2^X \setminus \{\emptyset\}$$

Le nombre K de classes doit nécessairement être déterminé à partir d'un sous-ensemble de X. Nous détaillons ce point par la suite.

Notons également qu'il est possible que certaines classes ainsi formées soient des classes atomiques ne comportant qu'une seule donnée de l'espace d'entrée. Dans ce cas particulier, nous emploierons abusivement le terme de donnée « non classée » puisque celle-ci n'est rapprochée d'aucune autre donnée (i.e., n'appartient pas à une classe regroupante).

6.1.3 Classification

Introduisons maintenant la notion centrale de classification.

Définition 6.6 (*Classification*)

La classification est la tâche visant à établir une fonction de classement d'un ensemble d'entrée dans un ensemble de classes $W = \{w_1, ..., w_K\}$, connues a priori. De manière plus formelle, une tâche de classification consiste en l'élaboration d'une fonction de classement de l'ensemble d'entrée dans l'ensemble des parties de l'ensemble des classes.

$$h : X \rightarrow 2^W$$

Notons que cette définition de la classification diffère de celle proposée dans (CORNUÉJOLS et MICLET 2010) pour inclure une dimension importante dans nos travaux : la classification multi-classes non-mutuellement exclusives. Ce type de classification permet d'affecter plusieurs étiquettes à une même donnée de l'espace d'entrée, ce qui n'est pas le cas en classification multi-classes mutuellement exclusives.

Notons également que contrairement à la tâche de regroupement, nous n'excluons pas la partie vide de l'ensemble de sortie. Ce choix est motivé par la nécessité de pouvoir n'affecter aucune étiquette à une donnée de l'ensemble d'entrée. Comme pour la tâche de regroupement, dans ce cas particulier, nous emploierons le terme de donnée « non classée » puisque celle-ci n'appartient pas à une classe.

Définition 6.7 (*Classification conceptuelle mono-classe*)

Une classification conceptuelle mono-classe est un cas particulier de classification dans lequel l'ensemble des étiquettes est limité à deux valeurs :

$$W = \{VRAI, FAUX\}$$

avec la condition de cohérence sur h suivante :

$$\forall x \in X, h(x) \neq \{VRAI, FAUX\}$$

Dans le cadre de la classification multi-classes non mutuellement exclusives, cette définition peut être étendue à la classification de concepts non-mutuellement exclusifs.

Définition 6.8 (*Classification conceptuelle multi-classes non-mutuellement exclusives*)

Une classification conceptuelle multi-classes non-mutuellement exclusives est un cas particulier de classification dans lequel l'ensemble des étiquettes est de taille $2 \times C$, où C est le nombre de concepts non-mutuellement exclusifs à étiqueter :

$$W = \{VRAI_i, FAUX_i\}_{1 \leq i \leq C}$$

avec la condition de cohérence sur h suivante :

$$\forall x \in X, \forall i \in [\![1, C]\!], \{VRAI_i, FAUX_i\} \not\subseteq h(x)$$

6.2 Apprentissage artificiel pour la classification et le regroupement

De manière non exhaustive, la réalisation des tâches de classification et regroupement peut être entreprise par apprentissage artificiel.

Définition 6.9 (*Algorithme d'apprentissage artificiel*)

Un algorithme d'apprentissage artificiel est un procédé visant à adapter des paramètres à la réalisation d'une tâche. De manière plus formelle, l'apprentissage artificiel a pour but dans nos travaux de déterminer par algorithme (i.e, automatiquement) une fonction de classement. La fonction de classement est dite apprise

et nous rappelons qu'elle possède le profil suivant :

$$h : X \rightarrow Y$$

L'apprentissage de la fonction de classement s'effectue soit à partir de la connaissance d'un sous-ensemble de X uniquement (apprentissage non supervisé), soit à partir de la connaissance d'un sous-ensemble de $X \times Y$ (apprentissage supervisé).

Nous rappelons que l'ensemble de sortie Y de la fonction de classement correspond respectivement aux ensembles $2^W \setminus \{\emptyset\}$ et 2^W en regroupements et classification.

Dans le cadre de la tâche de regroupement, nous ne disposons pas de connaissances précises sur Y. Nous pouvons uniquement faire des suppositions sur sa taille escomptée comme nous en avons déjà discuté auparavant. La seule connaissance précise est celle d'un sous-ensemble de X obtenu au travers d'exemples de textes à traiter. Nous disposons dans ce cadre des informations nécessaires pour mener un apprentissage non supervisé de la fonction de classement.

Dans le cadre de la tâche de classification, nous disposons de connaissances précises sur Y et sommes donc en mesure de déterminer un sous-ensemble de $X \times Y$. Nous disposons dans ce cadre des informations nécessaires pour mener un apprentissage supervisé de la fonction de classement.

L'apprentissage artificiel s'inspire de l'apprentissage naturel, réalisé par un humain. Il s'agit idéalement d'obtenir une adaptation à la tâche aussi bonne avec des techniques algorithmiques qu'avec les processus naturels. Ce but est loin d'être atteint et l'apprentissage artificiel se limite à des choix de modélisations mathématiques des tâches et à l'optimisation ou l'approximation de paramètres de ces modèles en vue de la réalisation de ces tâches.

Dans ce mémoire, notre but n'est pas de présenter exhaustivement les techniques utilisées en apprentissage artificiel mais uniquement celles utilisées pour la classification et le regroupement de textes. Nous ne présentons que les notions et méthodes utiles à cette fin.

6.2.1 Apprentissage non supervisé

Pour la tâche de regroupement, la fonction de classement est apprise en non supervisé à partir d'un sous-ensemble de X. Définissons formellement ce sous-ensemble.

Définition 6.10 (*Échantillon d'apprentissage en apprentissage non supervisé*)

Un échantillon d'apprentissage en apprentissage non supervisé correspond à un sous-ensemble de taille m de l'espace d'entrées X.

$$S = \{(x_i)\}_{[\![1,m]\!]}$$

Définissons à présent la notion d'algorithme d'apprentissage non supervisé.

Définition 6.11 (*Algorithme d'apprentissage non supervisé*)

Un algorithme d'apprentissage non supervisé identifie des régularités sous-jacentes à un échantillon d'apprentissage. Formellement, il s'agit de découvrir l'application h uniquement à partir de la connaissance d'un sous-ensemble de X de taille m. Cet algorithme est représenté formellement par :

$$\begin{aligned} A : \quad & 2^X \to H \\ & \textstyle\bigcup_{m \text{ fini}} \{S \sqsubseteq X \mid |S| = m\} \mapsto h \end{aligned} \tag{6.2}$$

avec $H = (2^{2^X \setminus \{\emptyset\}} \setminus \emptyset)^X$. Comme nous l'avons déjà explicité, lorsque H devient trop grand pour être exploré dans son intégralité, nous réduisons l'exploration des possibilité à un sous-ensemble $H = (2^W \setminus \{\emptyset\})^X$. En regroupement de textes, la taille très importante de l'ensemble X rend l'exploration de ce sous-ensemble de H encore difficile. Pour contourner ce problème, le choix standard est de considérer que le sous-ensemble S de X est suffisamment représentatif de X pour les substituer lors de l'apprentissage. Le sous-ensemble H à explorer est ainsi réduit à $H = (2^W \setminus \{\emptyset\})^S$.

6.2.2 Apprentissage supervisé

Pour la tâche de classification, la fonction de classement est apprise en supervisé à partir d'un sous-ensemble de $X \times Y$. Définissons formellement ce sous-ensemble.

Définition 6.12 (*Échantillon d'apprentissage en apprentissage supervisé*)

Un échantillon d'apprentissage en apprentissage supervisé correspond à un ensemble donnant des exemples déjà classés. Formellement, un échantillon d'apprentissage est un ensemble S de couples $(x, y) \in X \times Y$ de taille m.

$$S = \{(x_i, y_i))\}_{[\![1,m]\!]}$$

Définissons à présent la notion d'algorithme d'apprentissage supervisé.

Définition 6.13 (*Algorithme d'apprentissage supervisé*)

*Un algorithme d'apprentissage supervisé identifie une loi de dépendance sous-jacente à un échantillon d'apprentissage. Formellement, il s'agit de découvrir l'application h en cherchant une dépendance dans les couples (x, y), sous-ensemble de $X \times Y$ de taille m. L'apprentissage supervisé est définie dans (*CORNUÉJOLS *et* MICLET *2010, p 61) :*

$$A : \bigcup_{m=1}^{\infty} (X \times Y)^m \to H$$

Nous proposons la formule suivante :

$$A : \quad 2^{X \times Y} \to H$$
$$\bigcup_{m \text{ fini}} \{S \sqsubseteq X \times Y \mid |S| = m\} \to H$$

avec $H = Y^X$. En apprentissage de classification, l'ensemble de sortie de la fonction de classement équivaut à 2^W, ce qui donne $H = \left(2^W\right)^X$.

6.2.3 Discussions

L'apprentissage supervisé détermine la fonction h en ayant une connaissance précise de l'ensemble de sortie Y grâce à l'échantillon d'apprentissage donnant des valeurs de l'espace de sortie. Les tâches d'apprentissage supervisé, telles que l'apprentissage d'une classification, présupposent que l'échantillon d'apprentissage contient toutes les valeurs admissibles de l'espace de sortie, donnant ainsi une connaissance totale sur cet ensemble. A l'inverse, les tâches d'apprentissage non supervisé, telles l'apprentissage de regroupements, n'ont que peu de connaissances *a priori* sur cet espace.

Notons que les définitions données pour l'apprentissage supervisé et non supervisé ne fixent pas d'objectifs en terme de qualité de la fonction h apprise. Nous discutons plus loin de l'évaluation de l'apprentissage par l'évaluation de la fonction h de classement.

6.3 Ensembles d'entrée et d'hypothèses

6.3.1 Ensemble des hypothèses

Que ce soit en apprentissage supervisé ou en apprentissage non supervisé, l'algorithme doit déterminer une application h de l'espace d'entrée dans l'espace de sortie en ayant une connaissance limitée de l'espace d'entrée. Cela constitue un premier problème pour l'algorithme d'apprentissage.

Il doit apprendre à décider sur des éléments qui lui sont inconnus au moment de l'apprentissage de la fonction h. Cela implique nécessairement l'introduction d'une induction dans l'apprentissage sous peine d'obtenir au final une fonction h ne pouvant décider que les exemples vus lors de l'apprentissage (apprentissage par cœur). Comment l'algorithme peut-il généraliser la connaissance partielle qu'il a du problème ? L'algorithme doit limiter son ensemble d'hypothèses initiales en ne retenant que celles en adéquation avec certaines suppositions faites sur la tâche à apprendre. Ces réflexions conduisent à définir la notion de biais d'apprentissage.

Définition 6.14 (*Biais d'apprentissage*)

Un biais d'apprentissage est une restriction faite sur l'ensemble des hypothèses H.

Sans cette restriction, l'ensemble des hypothèses serait trop grand et surtout trop peu connu pour que l'algorithme d'apprentissage puisse trouver une fonction h adéquate. En réduisant judicieusement l'espace, l'algorithme peut alors parvenir à effectuer une recherche efficiente sur cet espace. L'introduction du biais permet en outre d'imposer une généralisation ou une abstraction de l'apprentissage par rapport aux données de l'échantillon d'apprentissage.

6.3.2 Ensemble d'entrée

Nous discutons à présent de l'ensemble d'entrée d'abord dans le cas général puis dans le cas des textes.

Discussions générales

L'ensemble d'entrée peut être exprimé au moyen de langages plus ou moins complexe de représentations. Parmi eux, la représentation sous la forme d'un n-uplet de valeurs est très utilisée :

Définition 6.15 (*Ensemble d'entrée*)

L'ensemble d'entrée est l'ensemble X. Chaque élément de X est caractérisé par les valeurs données à un ensemble d'attributs $\{x_1, ..., x_d\}$:

$$\forall x \in X, x = (x_1, ..., x_d)$$

Intéressons-nous un peu plus en détails aux propriétés de ces attributs :

1. Le nombre d d'attributs utilisés pour construire la représentation doit être le plus restreint possible. Cette condition permet de faciliter l'induction réalisée par les algorithmes d'apprentissage. Les algorithmes d'apprentissage ne parviennent pas à de bonnes performances inductives lorsqu'on leur soumet un problème contenant une quantité trop importante d'attributs : ce phénomène

est connu sous le nom de « malédiction des hautes dimensions ». Cette « malédiction » caractérisée également par une « perte des données dans l'espace » est décrite notamment dans (ERTÖZ, STEINBACH et KUMAR 2003).

2. Les différents attributs doivent être indépendants. En d'autres termes, la valeur prise par un attribut ne doit pas être déductible de la valeur prise par d'autres attributs. Dans le cas contraire, cet attribut n'apporte aucune information supplémentaire et doit être écarté pour limiter le nombre d'attributs.
3. Les attributs peuvent être de différents types : il peut s'agir d'attributs numériques ou encore d'attributs nominaux. Des attributs booléens peuvent également être employés. Cependant, la plupart des algorithmes d'apprentissage n'acceptent pas les représentations avec des attributs de natures différentes. On parle dans ce cas de représentations mixtes en opposition aux représentations homogènes. Dans le cas où l'on doit utiliser des représentations mixtes, il est usuel de transformer les attributs n'étant pas de bonne nature pour rendre homogène la représentation.

Notons que dans le cas d'attributs à valeurs numériques, la représentation de X est vectorielle.

Beaucoup de biais d'apprentissage mettent en jeu une relation de similarité ou dissimilarité sur l'ensemble d'entrée. Lorsque l'ensemble d'entrée est muni d'une telle relation, il est désigné sous le nom d'espace d'entrées ou encore espace de représentations.

Définition 6.16 (*Espace de représentations*)

Un espace de représentations ou espace d'entrée est un ensemble d'entrées telle qu'une relation de dissimilarité $diss$ puisse être établie entre les éléments de l'ensemble.

$$\begin{aligned} diss \;:\; & X \times X \to \mathbb{R}^+ \\ & (x_1,...,x_d),(x'_1,...,x'_d) \mapsto diss((x_1,...,x_d),(x'_1,...,x'_d)) \end{aligned}$$

La relation est définie selon la nature des attributs. S'il s'agit d'attributs numériques, la relation de dissimilarité peut être vectorielle. Dans le cadre d'attributs nominaux, la relation de dissimilarité est souvent définie de manière explicite par exemple

par une arborescence. Pour des attributs booléens, une dissimilarité simple s'appuyant sur leur valeur de vérité peut être utilisée. Les types d'attributs les plus naturels pour l'apprentissage sont les attributs booléens et numériques, les attributs nominaux pouvant la plupart du temps être convertis facilement dans l'un de ces deux types.

Espace de représentation de textes

Dans nos travaux, nous nous intéressons aux regroupements et aux classifications de textes. L'ensemble de représentation des textes est donc celui étudié en tant qu'ensemble d'entrée d'une fonction de classement h. L'apprentissage repose sur la découverte d'une application h permettant de classer n'importe quelle donnée de l'ensemble d'entrée à partir seulement de la connaissance d'un échantillon limité de cet ensemble.

Si les textes sont représentés de manière triviale d'un point de vue informatique en tant que suite de caractères, l'ensemble d'entrées atteint une taille beaucoup trop grande pour qu'un apprentissage puisse être réalisé avec seulement quelques dizaines de milliers d'exemples. Ce cas est un exemple extrême d'un espace d'entrées construit à partir d'un nombre d'attributs trop important, soumis à la malédiction des hautes dimensions que nous avons évoquée.

Outre la malédiction des hautes dimensions, la représentation de textes triviale ne permet pas d'établir une dissimilarité efficace entre les textes selon leur sémantique. Seules des dissimilarités d'édition peuvent être établies, ce qui n'est guère pertinent dans le cadre de classements thématiques de textes.

C'est pourquoi nous devons avoir recours à une représentation de ces textes telles que celles que nous avons présentées au chapitre 3 et à des relations de similarité ou dissimilarité sur ces dernières telles que celles que nous avons présentées au chapitre 2 afin de pouvoir utiliser avec profit des techniques d'apprentissage dans le cadre de textes.

Chapitre 7

Apprentissage artificiel supervisé : classification de textes

Nous présentons plus en détails dans ce chapitre l'apprentissage supervisé dans le cadre de classification de textes.

7.1 Objectif et évaluation de la classification

Notre objectif est d'établir une loi de dépendance (fonction de classement h) entre un espace d'entrée X et un espace de sortie Y en vue de la classification de textes.

$$h : X \rightarrow Y$$

En classification de textes, l'ensemble d'entrée X est celui des représentations des textes obtenus à partir des symboles de la langue. L'ensemble de sortie Y correspond à l'ensemble constitué des différentes étiquettes dénotant chacune une classe :

$$Y = 2^W$$

L'apprentissage supervisé est utilisé afin de découvrir la loi h depuis un échantillon d'apprentissage.

7.1.1 Objectif qualitatif et évaluation de l'apprentissage

Introduisons d'abord la notion d'oracle.

Définition 7.1 (*Oracle*)

Un oracle permet d'obtenir la prédiction de la sortie pour une donnée en entrée. Cette prédiction est supposée réalisée par un « expert » dans la tâche et est donc non erronée. De manière plus formelle, on suppose qu'un oracle est une application f définie de X dans Y telle que :

$$\begin{aligned} f \quad : \quad & X \to Y \\ & x \mapsto f(x) \end{aligned}$$

La nécessité de l'apprentissage réside dans le coût induit par le recours à l'oracle. Ce coût est supposé trop grand pour que l'oracle puisse réaliser la tâche de prédiction sur l'ensemble des éléments de X. Le recours à l'oracle ne peut alors se faire que sur un sous-ensemble de X, noté conventionnellement S et désigné par échantillon d'apprentissage.

Le but de l'apprentissage est alors de découvrir, grâce à S, la fonction f utilisée par l'oracle ou en d'autres termes d'apprendre une fonction h tel que :

$$\forall x \in X, h(x) = f(x)$$

De manière plus modeste et surtout plus réaliste, le système apprenant découvre une fonction h minimisant l'écart de classification avec la fonction f de l'oracle :

$$\forall x \in X, h(x) \approx f(x)$$

La relation $\approx$ définie ici entre la classification de l'oracle et celle du système apprenant nous permet d'introduire, de manière simple, la notion d'erreur de classifi-

cation du système. Cette notion est précisée plus loin dans ce chapitre.

L'étude de l'erreur de classification est motivée par deux points dans nos travaux :

1. Pour que l'approximation soit parfaite, il faut supposer que le biais choisi pour l'apprentissage l'est. Dans des conditions réelles de classification en langage naturel, il est illusoire de le supposer. Nous sommes donc nécessairement confrontés à des erreurs de classification.
2. L'application f fournie par l'oracle n'est pas une application idéale dans nos travaux. La construction de l'échantillon d'apprentissage est réalisée semimanuellement, durant une longue période et par des personnes différentes. Il est fort peu probable que l'étiquetage soit en tous points cohérent au fil du temps. Une même donnée de l'espace d'entrée peut alors être étiquetée par plusieurs sorties différentes en fonction de variations dans la qualité de l'étiquetage manuel des données. Dans ces conditions, la fonction f ne peut être considérée comme idéale (i.e., elle peut engendrer des erreurs de classification).

La définition donnée précédemment pour l'apprentissage supervisé ne donne pas d'objectifs à atteindre en terme qualitatif sur l'hypothèse h retenue par l'apprenant. Il semble naturel d'évaluer cette hypothèse par rapport à ses performances en terme de classification de nouvelles données de l'espace d'entrée. Les réflexions énoncées au préalable nous ont ainsi permis de poser que $\forall x \in X, h(x) \approx f(x)$. Le but ici est de définir des méthodes permettant d'évaluer en quoi une hypothèse h_1 est meilleure qu'une hypothèse h_2.

Tout d'abord, nous devons définir une fonction permettant de quantifier et pénaliser les erreurs commises par la fonction h par rapport à la fonction f. C'est là le but de la fonction de perte.

Définition 7.2 (*Fonction de perte*)

Une fonction de perte est une application l_h définie de l'ensemble $X \times Y$ dans $\mathbb{R}^+$ en fonction d'une application de décision h par :

$$\begin{aligned} l_h \quad : \quad & X \times Y \to \mathbb{R}^+ \\ & (x, y) \mapsto l(h(x), y) \end{aligned}$$

avec $\forall h \in H$,

$$\begin{aligned} l \quad : \quad & Y \times Y \to \mathbb{R}^+ \\ & (y, y') \mapsto l(y, y') \end{aligned}$$

La fonction de perte 0/1 est souvent employée en classification pour sa relative simplicité.

Définition 7.3 (*Fonction de perte 0/1*)

La fonction de perte 0/1 est la fonction de perte définie par :

$$l(h(x), y) = \begin{cases} 0 \text{ si } h(x) = y \\ 1 \text{ si } h(x) \neq y \end{cases}$$

Grâce à la fonction de perte, nous pouvons définir le risque réel, correspondant à la perte réelle engendrée par le choix d'une hypothèse h par rapport à la classification idéale réalisée par la fonction f dans un environnement suivant une loi de probabilité nommée P_{XY}.

Définition 7.4 (*Risque réel*)

Le risque réel est une application de l'ensemble H dans $\mathbb{R}^+$ *telle que :*

$$\begin{aligned} R_{Réel} \quad : \quad & H \to \mathbb{R}^+ \\ & h \mapsto \int_{x \in X, y \in Y} l(h(x), y) P_{XY} \, \mathrm{d}x \, \mathrm{d}y \end{aligned}$$

Le risque réel permet de définir la fonction de décision optimale, notée h^*.

Définition 7.5 (*Fonction de décision optimale*)

La fonction de décision optimale dans un ensemble H est l'hypothèse h minimisant le risque réel. Soit :

$$h^* = \underset{h \in H}{\arg\min} \, R_{Réel}(h)$$

La fonction h^* est la meilleure fonction h que l'apprenant puisse apprendre parmi l'ensemble des hypothèses H retenues par le biais choisi. D'une manière plus générale, établir qu'une hypothèse h_1 est meilleure qu'une hypothèse h_2 peut se formaliser par un pré-ordre sur l'ensemble des hypothèses H, noté $\geq^H_{Réel}$. Cette relation de pré-ordre peut être établie grâce au risque réel :

$$h_1 \geq^H_{Réel} h_2 \Leftrightarrow R_{Réel}(h1) \leq R_{Réel}(h2)$$

Le calcul de l'hypothèse h^* ne peut se faire via le calcul du risque réel car celui-ci est dans la pratique incalculable. Nous avons vu au préalable que nous n'avons qu'une connaissance partielle de X au travers de l'échantillon d'apprentissage. La distribution de probabilité P_{XY} ne peut donc être connue de manière parfaite. Il faut donc l'estimer. L'échantillon d'apprentissage peut être utilisé dans ce sens.

La distribution estimée « simple » de P_{XY} considère tous les couples (x_i, y_i) de S comme équi-probables de probabilité $\frac{1}{m}$. Tous les autres couples de $X \times Y$ ont une probabilité nulle.

Nous calculons ainsi la perte empirique réalisée par le choix d'une hypothèse h grâce au calcul du risque empirique.

Définition 7.6 (*Risque empirique*)

Le risque empirique est une application de H dans $\mathbb{R}^+$ utilisant l'échantillon d'apprentissage $S = \{(x_i, y_i)\}_{1,m}$ tel que :

$$\begin{aligned} R_{Emp} \quad : \quad & H \to \mathbb{R}^+ \\ & h \mapsto \frac{1}{m} \sum_{i=1}^{m} l(h(x_i), y_i) \end{aligned}$$

Le risque empirique permet d'obtenir l'hypothèse optimale au sens de la Minimisation du Risque Empirique (MRE). L'hypothèse étant obtenue au travers de l'échantillon d'apprentissage S, nous la notons h^*_S.

Définition 7.7 (*Principe de Minimisation du Risque Empirique (MRE)*)

Le principe de Minimisation du Risque Empirique propose de retenir dans un ensemble d'hypothèses H celle minimisant le risque empirique lorsque S est l'échan-

tillon d'apprentissage.

$$h_s^* = \arg\min_{h \in H} R_{Emp}(h)$$

Cela nous conduit à définir une relation de pré-ordre sur l'ensemble des hypothèses selon le risque empirique :

$$h_1 \geq_{Emp}^{H} h_2 \Leftrightarrow R_{Emp}(h_1) \leq R_{Emp}(h_2)$$

Les fonctions f, h^* et h_S^* ne coïncident pas en général. Elles représentent la fonction de décision optimale selon plusieurs niveaux de critères pris en compte. Ainsi, f représente la fonction optimale *a priori* de choix restrictifs sur l'espace d'hypothèses. La fonction h^* donne quant à elle la fonction optimale *a posteriori* de choix restrictifs sur l'espace d'hypothèses conduisant à ne retenir que l'ensemble H. Enfin, la fonction h_S^* représente la décision optimale sur l'ensemble H en ne retenant que les exemples de l'échantillon d'apprentissage. Les différences de risque engendrées par ces trois visions du but à atteindre par l'apprentissage donnent les erreurs d'approximation, d'estimation et totale.

Définition 7.8 (*Erreur d'approximation*)

L'erreur d'approximation correspond à la différence de risque engendrée par les choix restrictifs de l'espace des hypothèses à l'ensemble H.

$$E_{Approx} = R_{Réel}(h^*) - R_{Réel}(f)$$

Définition 7.9 (*Erreur d'estimation*)

L'erreur d'estimation correspond à la différence de risque engendrée par l'utilisation de l'échantillon d'apprentissage pour en extraire une quantité empirique en remplacement de la distribution de probabilités inconnue.

$$E_{Estim} = R_{Réel}(h_S^*) - R_{Réel}(h^*)$$

Définition 7.10 (*Erreur totale*)

L'erreur totale correspond à la différence de risque entre l'hypothèse optimale engendrée par l'utilisation de l'échantillon d'apprentissage pour en extraire une quantité empirique en remplacement de la distribution de probabilités inconnue et l'hypothèse optimale a priori de choix restrictifs sur l'ensemble des hypothèses. Il s'agit donc du cumul des erreurs d'approximation et d'estimation.

$$\begin{aligned} E_{Totale} &= R_{Réel}(h_S^*) - R_{Réel}(f) \\ &= R_{Réel}(h_S^*) - R_{Réel}(h^*) + R_{Réel}(h^*) - R_{Réel}(f) \\ &= E_{Estim} + E_{Approx} \end{aligned}$$

Le risque réel de la fonction f est nul si celle-ci correspond de manière parfaite à la fonction de décision recherchée. Dans le cas où les informations ne sont pas connues de manière parfaite, comme c'est le cas dans nos travaux avec l'échantillon d'apprentissage qui est bruité, le risque réel de f n'est pas nul.

7.1.2 Évaluation dans le domaine de la recherche d'informations

Dans certains domaines d'applications, les mesures du risque réel ou du risque empirique ne sont pas les plus naturelles à employer. C'est le cas dans le domaine de la recherche d'informations.

Définition 7.11 (*Recherche d'informations*)

Pour un ensemble d'informations I et une requête Q, le problème de recherche d'informations identifie dans l'ensemble I, les informations pertinentes pour la requête Q. Formellement, pour une requête Q, la recherche d'informations correspond à une application de I dans $\{P,N\}$ tel que :

$$\begin{aligned} IR(Q) \quad : \quad & I \rightarrow \{P,N\} \\ & i \mapsto \begin{cases} P \text{ ssi } i \text{ est pertinente pour } Q \\ N \text{ ssi } i \text{ n'est pas pertinente pour } Q \end{cases} \end{aligned}$$

Classe cible / **Classe estimée**	P	N
P	Vrais Positifs (VP)	Faux Positifs (FP)
N	Faux Négatifs (FN)	Vrais Négatifs (VN)

TABLE 7.1 – Classes résultantes des classes formées par une recherche d'informations. Les colonnes donnent les deux classes cibles mutuellement exclusives et les lignes les deux classes estimées mutuellement exclusives. Chaque intersection donne le nom de la classe attribuée aux données issues de l'intersection des deux classes.

Le problème de la recherche d'informations s'apparente à une classification selon deux classes mutuellement exclusives « positive » et « négative ». En sachant que chaque information de I appartient à l'une des deux classes, une partition en quatre classes des éléments de I peut être formée selon le tableau 7.1 à partir des quatre classes initiales.

Il est ainsi usuel de distinguer les deux types d'erreurs que sont la découverte d'une information erronée (Faux Positifs) et l'absence de découverte d'une information (Faux Négatifs). De ces deux types d'erreurs découlent les deux critères d'évaluation que sont la précision et le rappel.

Définition 7.12 (*Précision*)

La précision est la mesure donnant la capacité du système à trouver uniquement des informations correctes par rapport à une recherche. Cette mesure sanctionne de ce fait les erreurs de découverte d'informations erronées.

$$P(IR,Q) = \frac{|VP|}{|VP| + |FP|}$$

Définition 7.13 (*Rappel*)

Le rappel est la mesure donnant la capacité du système à trouver l'ensemble des informations correctes par rapport à une recherche. Cette mesure sanctionne de ce fait les erreurs d'absence de découverte d'informations.

$$R(IR,Q) = \frac{|VP|}{|VP| + |FN|}$$

La précision et le rappel ne fournissent une information pertinente sur la qualité du système que lorsqu'ils sont étudiés ensemble. Dans le cas contraire, il est possible de maximiser l'un ou l'autre de manière indépendante en adoptant deux solutions triviales. Ainsi, un système renvoyant très peu de résultats positifs, c'est-à-dire ne prenant que très peu de risques sur les faux positifs, a une bonne précision alors qu'il écarte beaucoup d'informations pertinentes. A l'inverse, un système renvoyant presque toutes les informations disponibles en résultats positifs, c'est-à-dire ne prenant que très peu de risques sur les faux négatifs, a un bon rappel alors qu'il renvoie beaucoup d'informations non pertinentes.

Il est ainsi usuel d'avoir recours à une mesure agrégeant les deux critères présentés. Une agrégation basée sur la moyenne harmonique de la précision et du rappel est couramment choisie.

Définition 7.14 (*F-mesure*)

La F-mesure est donnée par une moyenne harmonique pondérée de la précision et du rappel. Le coefficient de pondération est noté β.

$$F\text{-}mesure(P,R) = \frac{(1+\beta^2).P.R}{\beta^2.P+R}$$

La valeur de β est souvent prise à 1. Dans ce cas, la F-mesure, notée F_1-mesure équivaut à une moyenne harmonique non pondérée de la précision et du rappel.

$$F_1\text{-}mesure(P,R) = \frac{2.P.R}{P+R}$$

Les notions de précision, rappel et F-mesure nous seront particulièrement utiles par la suite. Ils nous permettront d'évaluer concrètement les performances de classification des fonctions apprises par les différents algorithmes d'apprentissage supervisé. Nous consacrons la suite de ce chapitre à la présentation de ces algorithmes.

7.2 Algorithmes d'apprentissage supervisé

Nous détaillons quelques algorithmes utilisés pour l'apprentissage supervisé.

7.2.1 Classification bayésienne et approximation

Présentation empirique de la classification bayésienne et application à la classification

Soit S, l'échantillon d'apprentissage. Supposons que l'on connaisse la probabilité $P(S)$ d'apparition de l'échantillon d'apprentissage. Cette supposition n'est pas réaliste mais permet de poser une définition empirique de la classification bayésienne. Nous notons la probabilité d'une hypothèse $h \in H$ par $P(h)$ et la probabilité conditionnelle de h selon S par $P(h|S)$.

Définition 7.15 (*Règle de décision bayésienne optimale selon H*)

La classification bayésienne sélectionne l'hypothèse $h \in H$ maximisant la probabilité conditionnelle $P(h|S)$.

$$h^* = \arg\max_{h \in H} P(h|S)$$

Dans le cas où H est infini, il faut supposer l'existence d'une densité de probabilité $p(h)$ et d'une densité de probabilité conditionnelle $p(S|h)$. Nous pouvons ainsi calculer :

$$h^* = \arg\max_{h \in H} p(S|h)\,p(h)$$

Nous avons déjà précisé qu'il n'était pas possible d'obtenir la probabilité $P(S)$ dans les cas réels d'utilisation. De ce fait, cette décision optimale ne peut qu'être approchée. De nombreuses techniques existent pour cela. Nous ne présentons ici que les plus utiles à nos travaux.

Pour le problème de la classification, il s'agit d'identifier la classe w_i la plus probable pour $x \in X$ donné. Soit maximiser la probabilité de w_i sachant x :

$$P(w_i|x) = \frac{p(x|w_i)P(w_i)}{p(x)}$$

selon la règle de Bayes.

Pour toutes les classes considérées, la densité $p(x)$ est identique, positive et non nulle. La classe maximisant la probabilité $P(w_i|x)$ est donnée par :

$$w_i^* = \underset{w_i \in Y}{\arg\max}\, p(x|w_i)P(w_i)$$

Nous devons donc estimer la densité $p(x|w_i)$ et la probabilité $P(w_i)$ pour déterminer la meilleure classe.

Estimation locale de la densité de probabilités

Cette méthode vise à donner une estimation locale de la densité de probabilités de chacune des classes : $p(x|w_i)$.

Soient $p(x)$ une densité de probabilité et R une région où cette densité est définie. La probabilité pour qu'un point x tiré selon cette distribution de probabilité appartienne à R est donnée formellement par :

$$P = \int_R p(x)\,\mathrm{d}x$$

Soit un échantillon de m points tirés selon cette loi dont k points appartiennent à la région R. La loi des grands nombres nous permet de donner une approximation de P, sous la condition que m soit suffisamment grand :

$$P \simeq \frac{k}{m}$$

Il en découle que :

$$P = \int_R p(x)\,\mathrm{d}x \simeq \frac{k}{m}$$

Si l'on considère une région R suffisamment restreinte, ce qui est le cas pour une estimation locale de la densité de probabilité, on peut alors postuler que la densité de probabilité n'évolue que très peu sur cette région et peut donc être étudiée en tant que constante sur cet intervalle. Dans ces conditions, l'équation précédente devient :

$$P = \int_R p(x)\,\mathrm{d}x \simeq p(x)\int_R \mathrm{d}x \simeq p(x)V \simeq \frac{k}{m}$$

où V représente le volume de la région R.

On obtient alors :

$$p(x) \simeq \frac{\frac{k}{m}}{V}$$

A partir de ce résultat, nous pouvons estimer localement la densité de probabilité d'une classe w_i par l'intermédiaire de l'échantillon d'apprentissage. Soit m_i le nombre de points dans l'échantillon d'apprentissage appartenant à la classe w_i et k_i le nombre de points de l'échantillon d'apprentissage appartenant à la classe w_i et se situant dans une région R. La probabilité qu'un point x de la région R soit dans la classe w_i, notée $p(x|w_i)$, peut alors être approchée par l'équation :

$$p(x|w_i) \simeq \frac{\frac{k_i}{m_i}}{V}$$

D'un point de vue calculatoire, la densité de probabilité peut être approchée localement soit en fixant par avance la région R à partir du point où l'on cherche à l'estimer, soit en fixant par avance le nombre de points de l'échantillon d'apprentissage servant pour l'approximation.

Il est également possible d'avoir recours à des méthodes utilisant implicitement la règle de classification bayésienne sans toutefois nécessiter le calcul local de la den-

sité de probabilité. C'est le cas de la méthode dites des K plus proches voisins.

K plus proches voisins

L'algorithme des K plus proches voisins est l'algorithme de classement par supervision ayant un biais parmi les plus simples à mettre en œuvre. Il ne se base que sur une mesure de distance ou de similarité entre les données de l'échantillon d'apprentissage et les données à classer.

Le nombre K est un paramètre fixé en entrée de l'algorithme.

L'algorithme procède selon les étapes suivantes :

1. Calcul de la distance entre chaque donnée x de l'échantillon d'apprentissage et la nouvelle donnée x' à classer.
2. Pour les K plus proches données x de x', comptabilisation du nombre d'étiquettes y associées à chacun de ces x dans l'échantillon d'apprentissage
3. Affectation de l'étiquette y' à la nouvelle donnée correspondant à l'étiquette majoritaire sur l'ensemble des K étiquettes de l'étape 2.

En cas d'égalité sur les étiquettes majoritaires, une stratégie de départage est employée. On peut par exemple augmenter localement le nombre de voisins considérés jusqu'à levée de l'égalité.

L'algorithme 13 page C-11 donne la version en pseudo-code de cette méthode d'apprentissage.

Le choix de la valeur de K n'est pas anodin sur les décisions prises en terme de classification de nouvelles données. Comme l'illustre parfaitement la figure 7.1, la valeur de K peut être déterminante sur la classification adoptée. Ainsi, si $K = 3$, la nouvelle donnée symbolisée par la croix au centre de la figure sera classée en tant qu'étoile car deux plus proches voisins sur trois sont des étoiles. En revanche, si $K = 5$, la nouvelle donnée sera classée en tant que carré car 3 plus proches voisins sur 5 sont des carrés.

La méthode de classification des K plus proches voisins repose sur une estimation de la classification bayésienne et un classement des données selon la classe la plus

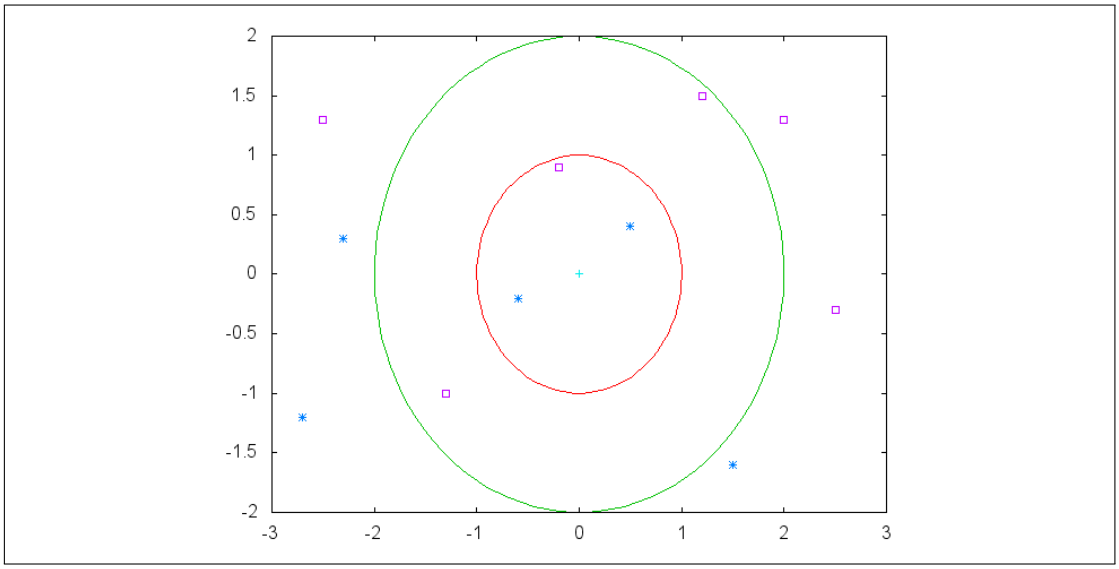


FIGURE 7.1 – Classification par les K plus proches voisins ($K = 3$ et $K = 5$). Le graphique comporte des éléments classés selon deux classes distinctes : la classe des étoiles bleues et la classe des carrés violets. L'élément central (croix bleue) est celui à classer. Les deux cercles symbolisent les limites de voisinage définies par les 3 et 5 plus proches voisins de l'élément central.

probable, c'est-à-dire ayant la densité de probabilité locale maximale en le point à classer.

Soient m_i le nombre de points dans l'échantillon d'apprentissage appartenant à la classe w_i, K le nombre de points de l'échantillon d'apprentissage dans le voisinage retenu par la méthode et k_i le nombre de points de l'échantillon d'apprentissage appartenant à la classe w_i dans le voisinage retenu par la méthode. La densité $p(x|w_i)$ s'exprime pour ce voisinage par :

$$p(x|w_i) \simeq \frac{\frac{k_i}{m_i}}{V}$$

D'autre part, la loi des grands nombres nous donne l'équation suivante :

$$P(w_i) \simeq \frac{m_i}{m}$$

On en déduit que :

$$k_i = mVp(x|w_i)P(w_i)$$

D'après la règle de Bayes, nous avons de plus,

$$P(x \cap w_i) = p(x|w_i)P(w_i) = P(w_i|x)p(x)$$

D'où,

$$k_i = mVP(x \cap w_i) = mVP(w_i|x)p(x)$$

Comme m, V et $p(x)$ sont constants, positifs et non nuls pour toutes les classes, cela revient à établir que k_i est proportionnel à $P(w_i|x)$ et qu'une classe maximisant k_i dans une région donnée maximise également la probabilité $P(w_i|x)$ dans cette même région. Cela démontre alors la validité bayésienne de cette approche sous la condition faite dans la démonstration que $P(w_i) \simeq \frac{m_i}{m}$.

7.2.2 Algorithmes à noyaux : passage par un espace de redescription

Dans de nombreux cas, il n'est pas possible d'effectuer une séparation des classes par une combinaison linéaire sur les attributs de l'espace d'entrée. Pourtant, il est intéressant de pouvoir optimiser le problème d'apprentissage par des techniques linéaire de par la simplicité calculatoire induite comparée à d'autres approches. Le but des méthodes à noyaux est de parvenir à cela par l'intermédiaire d'un espace de dimension très élevée équivalent grâce à une fonction de transfert ou encore fonction de redescription.

Définition 7.16 (*Fonction de transfert*)

La fonction de transfert ou fonction de redescription usuellement notée Φ permet de transformer une donnée x définie dans l'espace d'origine (espace source)

X en une donnée équivalente $\Phi(x)$ dans l'espace d'arrivée (espace cible) Y.

$$\begin{aligned} \Phi \quad : \quad & X \rightarrow Y \\ & x \mapsto \Phi(x) \end{aligned}$$

L'espace d'arrivée est couramment dénoté par espace de redescription. Le but recherché au travers de cette redescription des données est d'obtenir un espace dans lequel les données sont séparables linéairement. Le problème induit est la dimension potentiellement infinie de cet espace.

Soit un espace d'entrée. La fonction de transfert Φ sur les éléments de cet espace peut être décomposée en fonctions notés ϕ_i pour $i = 1, \infty$. Cela nous conduit à définir la fonction h recherchée par :

$$h(x) = \sum_{i=1}^{\infty} w_i \phi_i(x)$$

Notons ici la somme infinie qui nous empêche de réaliser ce calcul directement. La représentation duale du problème par le principe de minimisation du risque empirique nous donne cependant une autre formule pour la fonction h :

$$h(x) = \sum_{i=1}^{m} \alpha_i \phi_i(x_i)\phi_i(x) = \sum_{i=1}^{m} \alpha_i \Phi(x_i)\Phi(x) = 0$$

Définition 7.17 (*Fonction noyau*)

La fonction noyau permet de transformer les données dans le but d'obtenir un espace dans lequel les données sont séparables par un hyperplan pour la tâche désirée. La fonction noyau κ est définie entre deux données de l'espace d'origine x et x' selon la fonction de transfert Φ par :

$$\kappa(x, x') = \Phi(x).\Phi(x') = \sum_{i=1}^{\infty} \phi_i(x)\phi_i(x')$$

Cette définition nous permet alors de poser :

$$h(x) = \sum_{i=1}^{m} \alpha_i \kappa(x_i, x)$$

Citons quelques-unes des fonctions noyaux parmi les plus connues et les plus utilisées.

1. La fonction linéaire : $\kappa(x1, x2) = \gamma.x1.x2$
2. La fonction polynomiale : $\kappa(x1, x2) = (\gamma.x1.x2 + C_0)^n$ avec n, degré du polynôme et γ et C_0, les coefficients du polynôme.
3. La fonction sigmoïde : $\tanh(\gamma * x1 * x2 + C_0)$
4. La fonction basée sur le rayon (RBF) : $\exp(-\gamma * |x1 - x2|^2)$

Parmi les propriétés intéressantes, notons que l'utilisation de la fonction noyau linéaire avec un coefficient $\gamma = 1$ revient à considérer la fonction de transfert comme la fonction identité, soit $\Phi(x) = x$. L'espace des données n'est alors pas transformé.

Notons également que les paramètres des fonctions noyaux tel que le degré du polynôme pour la fonction polynomiale sont fixés en entrée de l'algorithme et doivent donc faire l'objet si besoin d'une optimisation indépendante.

Machines à vecteurs supports (SVM)

Parmi les algorithmes d'apprentissage supervisé utilisant les fonctions noyaux, l'algorithme des machines à vecteurs supports (SVM), aussi appelées séparateurs à vastes marges, est sans doute le plus largement utilisé.

L'algorithme identifie, lors de la phase d'entraînement, des vecteurs de données caractéristiques (i.e., décisifs) pour la tâche. Ces vecteurs supports délimitent un hyperplan dans l'espace de redescription, situé au plus loin des vecteurs supports (principe de la vaste marge) mais séparant les vecteurs supports des différentes catégories.

Cela permet lors de la classification, de déterminer la classe des données et la certitude de la classification selon leur position et leur éloignement par rapport à

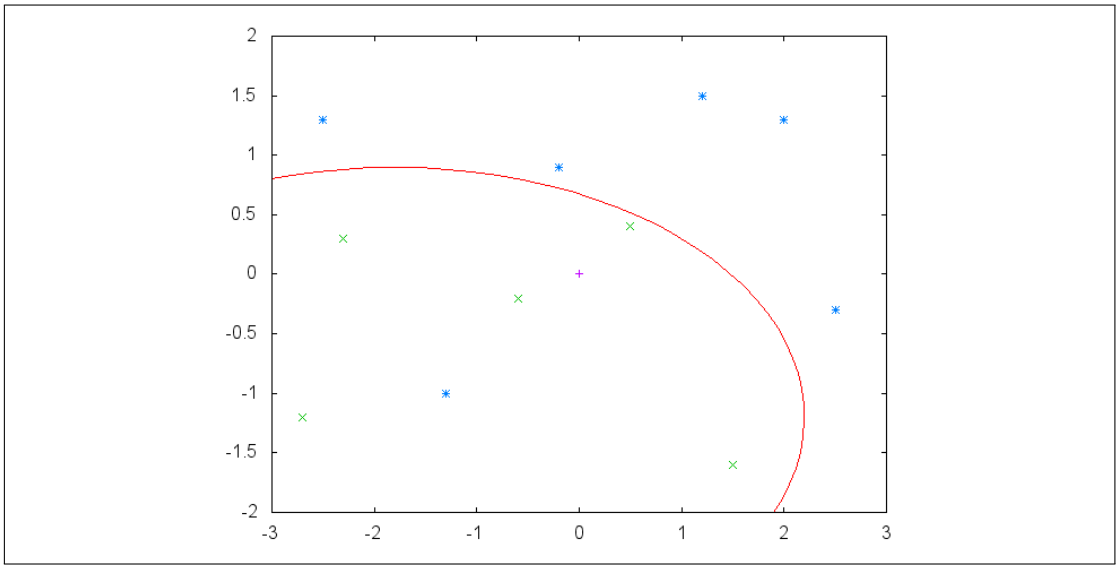


FIGURE 7.2 – Exemple de classification par les Machines à Vecteurs Supports (SVM). Le graphique comporte des éléments de deux classes distinctes : les étoiles bleues et les croix vertes. La croix violette au centre constitue l'élément à classer. La courbe symbolise l'hyperplan formé par l'algorithme SVM pour séparer les données des deux classes.

l'hyperplan. La figure 7.2 montre un exemple de délimiteur obtenu dans l'espace des données.

Notons qu'une des étoiles bleues n'a pu être correctement séparée. Dans ce cas, la marge obtenue est dite négative.

Chapitre 8

Apprentissage artificiel non supervisé : regroupements de textes

8.1 Objectif et évaluation de regroupements

Notre but est de découvrir des régularités (fonction de classement h) sous jacentes à des éléments issus d'un espace d'entrée X en vue de regroupements de textes dans un espace de sortie Y.

$$h : X \rightarrow Y$$

En regroupement de textes, l'ensemble d'entrée X est celui des représentations des textes obtenus à partir des symboles de la langue. L'ensemble de sortie Y correspond à l'ensemble des parties non vides de l'ensemble W constitué de K classes w_i à identifier sur X :

$$Y = 2^W \setminus \{\emptyset\}$$

L'apprentissage non supervisé est utilisé pour découvrir les régularités de l'espace d'entrée induites par h depuis un échantillon d'apprentissage.

8.1.1 Évaluations de l'ensemble de sortie

Définition 8.1 (*Centre de gravité d'une classe*)

Le centre de gravité d'une classe y constitué de m éléments $(x_1, x_2, ..., x_m)$, définis dans un espace d'entrée de d attributs tel qu'un élément $x_i = (x_{i1}, x_{i2}, ..., x_{id})$, est le point noté $\mu(y)$ de l'espace d'entrée tel que :

$$\mu(y) = \left(\frac{1}{m} \sum_{i=1}^{m} x_{i1},\ ...,\ \frac{1}{m} \sum_{i=1}^{m} x_{id} \right)$$

Nous notons $\mu(y)$ la fonction donnant le centre de gravité μ d'une classe y selon cette formule.

Le centre de gravité est par définition l'élément qui minimise la somme de la distance euclidienne entre lui et chacun des éléments de la partie. A partir de la définition du centre de gravité, nous définissons le premier critère d'évaluation en apprentissage non supervisé qu'est la variance intra classe.

Définition 8.2 (*Variance intra classe*)

La variance intra classe, abrégé en variance, est une mesure définie sur une classe y. La variance intra classe de y est :

$$V(y) = \frac{1}{|y|} \sum_{i=1}^{|y|} (x_i - \mu(y))^2$$

A nombres de parties égaux, la somme des variances intra classes peut être utilisée comme critère discriminant pour évaluer deux ensembles de parties. Soit c, le nombre de classes formées. La somme T des variances intra classes d'un ensemble de parties est donnée par :

$$T = \sum_{i=1}^{c} V(y_i)$$

Afin de mesurer la qualité des régularités trouvées en plus de la variance intra

classes, il peut être intéressant d'exploiter les irrégularités entre les différentes classes. Cela est d'autant plus pertinent que les algorithmes que nous présentons par la suite se servent de cette irrégularité entre classes pour mieux apprendre à les distinguer. Cette irrégularité entre classes est parfois dénotée par variance inter classes. Nous évitons cette appellation pour ne pas risquer de confusion avec la variance présentée auparavant et préférons parler de dissimilarité ou distance entre classes.

Définition 8.3 (*Dissimilarité entre classes*)

Une dissimilarité entre classes est une relation de dissimilarité sur l'ensemble de sortie Y.

$$\begin{aligned} diss \quad &: \quad Y \times Y \rightarrow \mathbb{R}^+ \\ &\quad\quad y_1 \times y_2 \mapsto diss(y_1, y_2) \end{aligned}$$

Donnons à présent quelques dissimilarités usuelles entre classes.

Définition 8.4 (*Indice du lien simple*)

L'indice du lien simple est la dissimilarité entre deux parties calculée comme étant la dissimilarité la plus faible existante entre deux éléments appartenant à l'une et l'autre des parties. C'est donc la dissimilarité entre les deux éléments les plus proches entre les deux parties.

$$diss(y_1, y_2) = \min_{x_1 \in y_1, x_2 \in y_2} diss(x_1, x_2)$$

Définition 8.5 (*Indice de distance entre centres de gravité*)

L'indice de distance entre centres de gravité est la dissimilarité entre deux parties calculée comme étant la dissimilarité existant entre les deux centres de gravité.

$$diss(y_1, y_2) = diss(\mu(y_1), \mu(y_2))$$

Ces notions sont utiles pour évaluer les regroupements effectués par les fonctions issus des apprentissages non supervisés. Nous consacrons la suite de ce chapitre à la présentation des algorithmes d'apprentissage non supervisé utiles pour nos travaux.

8.2 Algorithmes d'apprentissage non supervisé

8.2.1 Apprentissage par partitions : K-moyennes (*K-means*)

L'apprentissage par l'algorithme des K-moyennes (*K-means*) (LLOYD 1982) effectue l'apprentissage d'une partition de taille K sur un ensemble de données en minimisant la variance des classes par le biais de leurs centres de gravité.

Soient X l'ensemble des données, K le nombre de parties à apprendre et $\pi = \{P_1, P_2, ..., P_K\}$ la partition produite constituée de K parties. Soit M l'ensemble associant l'ensemble des K parties, notés P_i à un centre de gravité noté μ_i :

$$M = \{(P_1, \mu_1), (P_2, \mu_2), ..., (P_K, \mu_K)\}$$

L'algorithme des K-moyennes sélectionne une partition qui optimise l'emplacement des centres de gravité tel que :

$$\arg\min_{\pi} \sum_{i=1}^{K} \sum_{x_j \in X \cap P_i} ||x_j - \mu_i||^2$$

L'algorithme procède selon les étapes suivantes :

1. Initialiser pseudo-aléatoirement les K centres de gravité dans l'espace des données
2. Associer chaque donnée $x \in X$ à la partie P_i tel que le centre de gravité μ_i soit le plus proche des centres de gravité par rapport à x.
3. Recalculer les attributs de chaque μ_i pour qu'ils soient définis comme les moyennes des données contenues dans chaque partie P_i
4. Boucler sur l'étape 2 tant que la position d'au moins un des centres de gravité a changé à l'étape 3.

L'algorithme 10 page C-8 donne la version en pseudo-code de cet apprentissage.

La figure 8.1 donne un exemple de points à regrouper dans l'espace d'entrée.

En prenant $K = 2$, L'algorithme des K-moyennes peut se stabiliser sur la partition en deux parties représentée à la figure 8.2. Les croix rouges et vertes représentent les

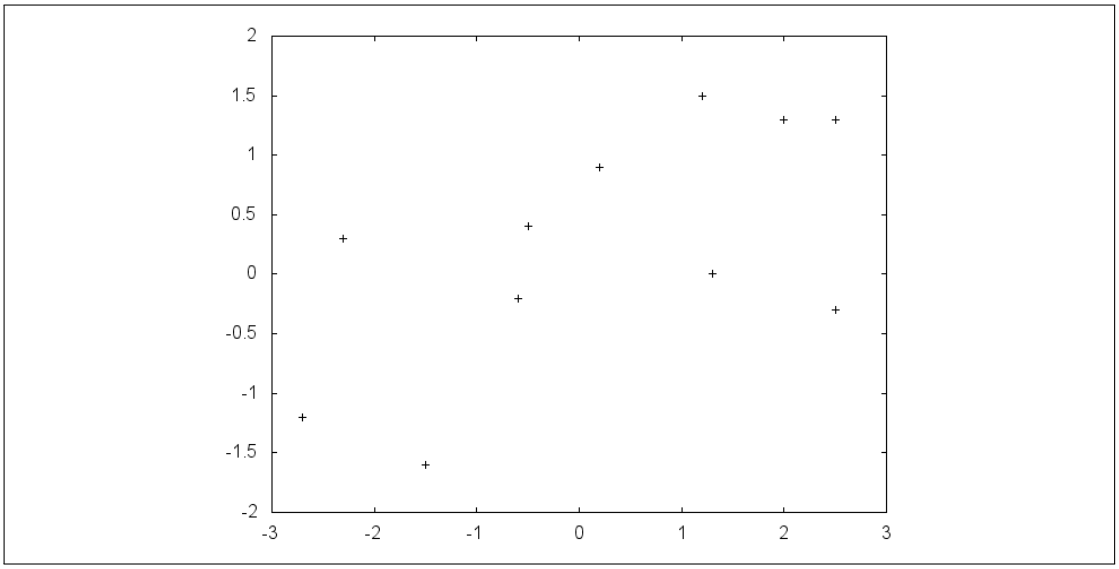


FIGURE 8.1 – Espace d'entrée de l'algorithme des K-moyennes. Les éléments du sous-ensemble d'apprentissage des éléments de cet espace sont représentés par des croix.

deux parties formées à partir des points de l'espace d'entrée. Les croix noires représentent les deux centres de gravité des parties.

Il est à noter que cet algorithme correspond à une méthode empirique d'optimisation de la variance de K classes formant une partition. En outre, l'initialisation aléatoire des K centres de gravité lors de la première phase de l'algorithme conditionne la convergence de ce dernier. Selon ce tirage aléatoire, l'algorithme peut se stabiliser sur une partition correspondant à un minimum local en terme de variance. Il n'y a donc aucune garantie de calculer avec cet algorithme une partition minimisant globalement la variance.

De plus, l'algorithme des K-moyennes demande de fixer le nombre de classes désiré en sortie au préalable de tout calcul. Cette contrainte est évidemment très forte et se révèle souvent un obstacle dans les applications concrètes de tâches de regroupement où l'on ne peut connaître à l'avance le nombre de classes optimal. Des algorithmes dérivés des K-moyennes ont donc été mis au point dans le but de rendre plus souple cette contrainte en permettant la création ou la suppression de classes pendant l'optimisation. La création ou la suppression de classes est décidé par des mesures telles que la variance intra classes ou la dissimilarité inter classes de manière à rendre plus ou moins fine la partition créée au final. Ces méthodes sont voisines des

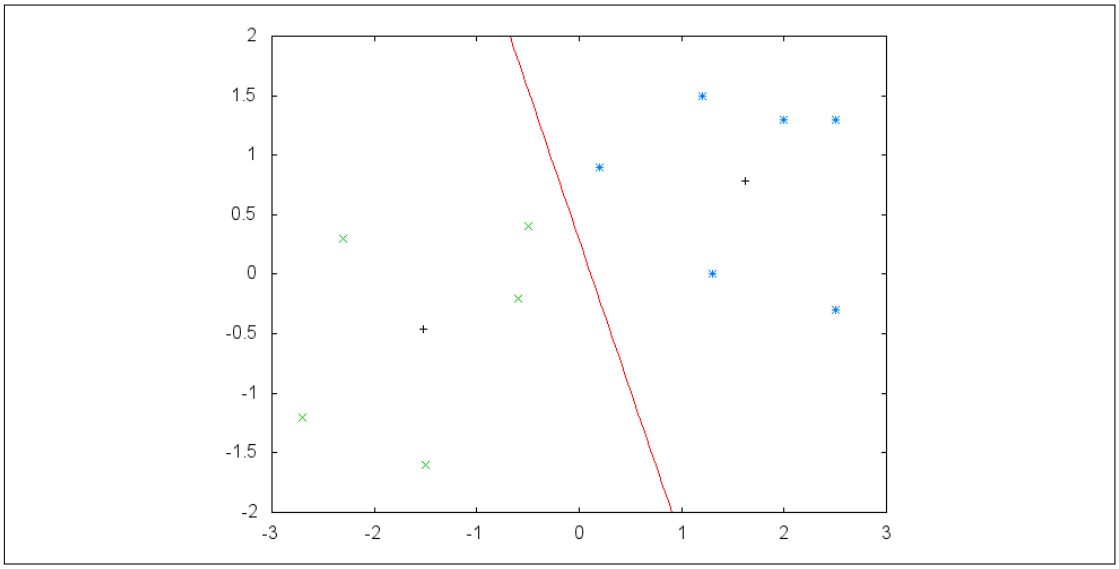


FIGURE 8.2 – Regroupements obtenus par les K-moyennes ($K = 2$). Les éléments d'apprentissage ont été regroupés selon une partition en deux classes : les éléments étoiles bleues et les éléments croix vertes. Les deux croix noirs représentent les centres de gravité des deux parties et la droite rouge la séparatrice des deux parties de la partition.

méthodes hiérarchiques que nous décrivons ci-après.

8.2.2 Apprentissage hiérarchique

La famille des algorithmes dits hiérarchiques regroupe deux techniques différentes mais corrélées : l'apprentissage hiérarchique ascendant et l'apprentissage hiérarchique descendant. Les deux techniques se basent sur la notion de hiérarchie de groupes.

Définition 8.6 (*Relation d'ordre de finesse entre partitions*)

Une partition π_i est au moins aussi fine qu'une partition π_j si et seulement si :

$$\forall P_i \in \pi_i, \exists P_j \in \pi_j, P_i \subseteq P_j$$

Définition 8.7 (*Hiérarchie de partitions*)

Pour tout ensemble de données $X = \{x_1, x_2, ..., x_m\}$, il est possible de former une hiérarchie de partitions H telle que celle-ci soit un ensemble de r partitions $\{\pi_1, \pi_2, ..., \pi_r\}$

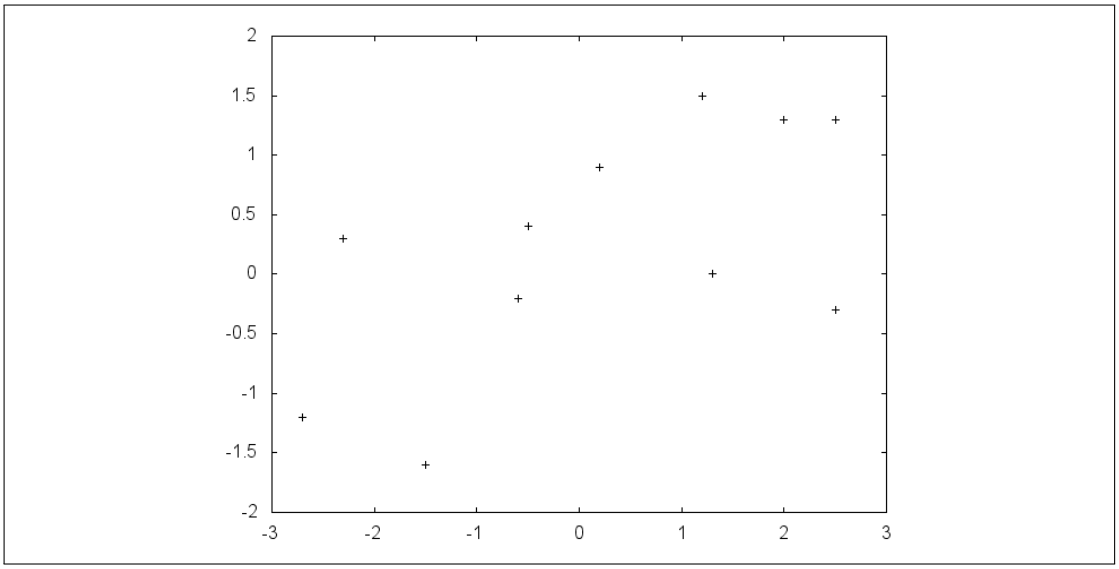


FIGURE 8.3 – Espace d'entrée des algorithmes hiérarchiques. Les éléments du sous-ensemble d'apprentissage des éléments de cet espace sont représentés par des croix.

tel que :

1. $\pi_1 = \{\{x_1, x_2, ..., x_m\}\}$ *est la partition la moins fine que l'on puisse créer avec l'ensemble de données* E.
2. $\pi_r = \{\{x_1\}, \{x_2\}, ..., \{x_m\}\}$ *est la partition la plus fine que l'on puisse créer avec l'ensemble de données* E.
3. $\forall i \in [\![1, r-1]\!], \pi_{i+1}$ *est plus fine que* π_i

On notera que l'utilisation de partitions garantit qu'il n'y a pas de chevauchements entre les groupes dans la hiérarchie. Concrètement, il existe deux possibilités pour construire une telle hiérarchie :

1. on commence avec π_1 et on affine $r-1$ fois la partition jusqu'à obtenir π_r.
2. on commence avec π_r et on désaffine $r-1$ fois la partition jusqu'à obtenir π_1.

Dans la pratique, les algorithmes d'apprentissage ne calculent pas les r partitions mais s'arrêtent lorsqu'ils ont atteint une partition π_i jugée d'une finesse acceptable en utilisant des critères d'homogénéité inter et intra parties.

Afin d'illustrer le fonctionnement des algorithmes d'apprentissage hiérarchique, prenons l'exemple de la figure 8.3 comme espace d'entrée

Apprentissage hiérarchique ascendant

Cette technique d'apprentissage regroupe les données en partant de la partition la plus fine et en remontant progressivement la hiérarchie vers la partition la moins fine jusqu'à obtenir une partition comportant des parties conformes à ce que l'on attend. L'union de deux parties dans la hiérarchie s'effectue selon le critère de proximité maximale entre ces deux parties par rapport aux autres. L'apprentissage ascendant procède comme suit :

1. Initialiser la hiérarchie avec la partition ayant le maximum de finesse : isolation de chaque élément de X dans sa propre partie.
2. Évaluer les distances entre chaque partie et fusionner les deux parties les plus proches pour former une nouvelle partition moins fine.
3. Boucler sur l'étape 2 tant qu'il existe une distance entre deux parties inférieure au seuil

L'algorithme 11 page C-9 donne la version en pseudo code de cet apprentissage.

Dans notre exemple, la partition la plus fine de l'espace est décrite à la figure 8.4. L'algorithme part de cette partition pour en former de moins fines jusqu'à atteindre le but escompté. Les figures 8.5, 8.6, 8.7, 8.8, 8.9 et 8.10 illustrent cette recherche d'une partition admissible.

A la fin de l'algorithme, la partition comporte 5 parties dont 4 non unitaires et 1 unitaire. Nous qualifions de « non classé » l'élément de cette partie unitaire.

Apprentissage hiérarchique descendant

Cette technique d'apprentissage part de la partition la moins fine regroupant toutes les données et la scinde jusqu'à obtenir une partition comportant des parties conformes à ce que l'on attend. L'algorithme descendant procède comme suit :

1. Initialiser la hiérarchie avec la partition ayant le minimum de finesse : regroupement de tous les éléments de X dans une partie unique.

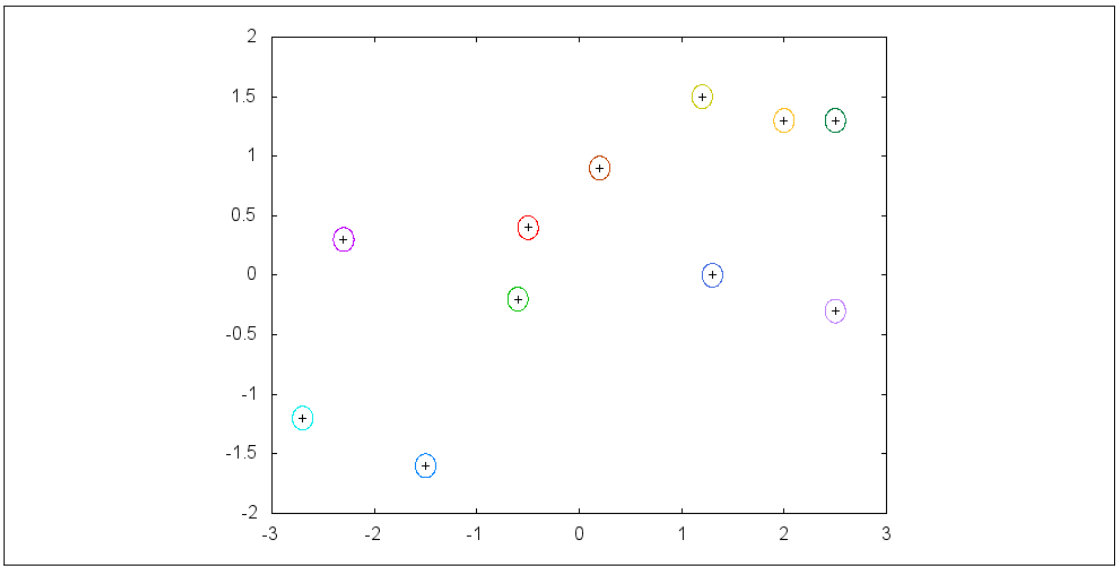


FIGURE 8.4 – Partition initiale de l'algorithme hiérarchique ascendant : la plus fine existante sur l'espace d'entrée. Les 11 parties sont unitaires ($K = 11$).

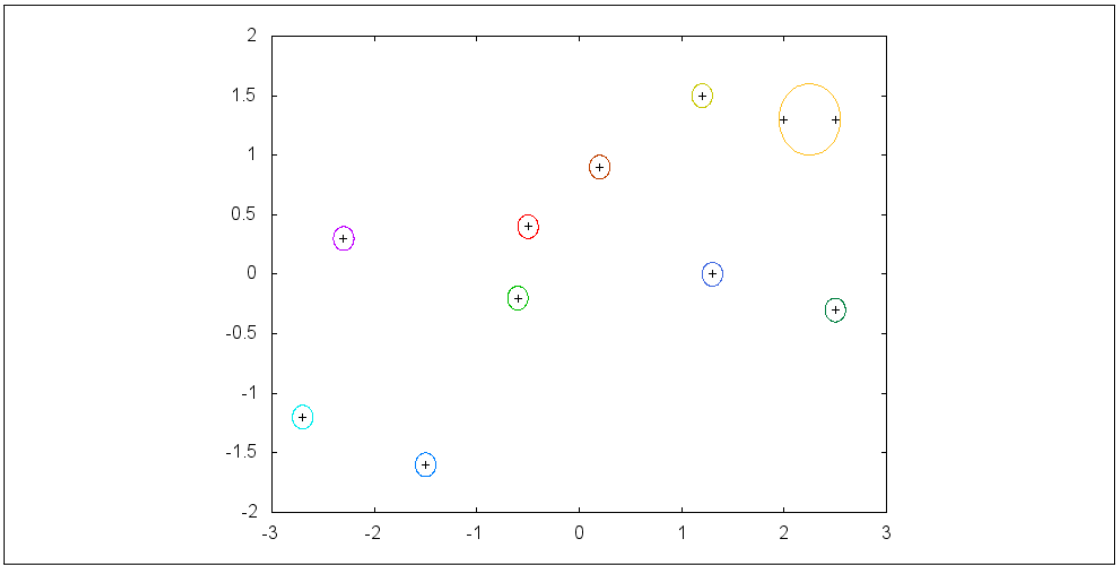


FIGURE 8.5 – Partition intermédiaire de l'algorithme hiérarchique ascendant : 1er regroupement. Une seule des 10 parties n'est plus unitaire ($K = 10$).

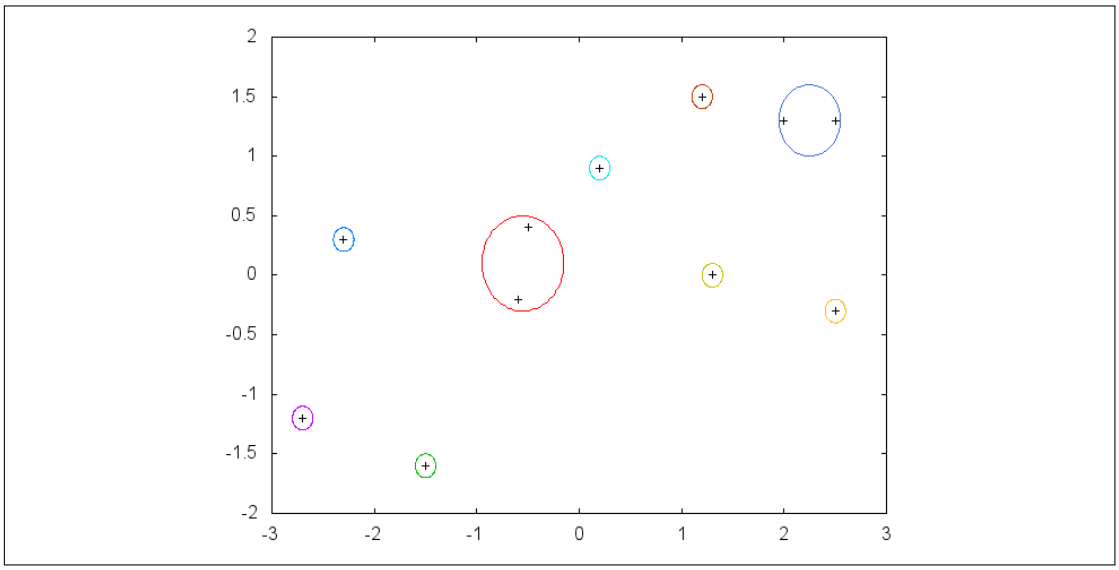


FIGURE 8.6 – Partition intermédiaire de l'algorithme hiérarchique ascendant : 2e regroupement. Deux parties sur 9 ne sont plus unitaires ($K = 9$).

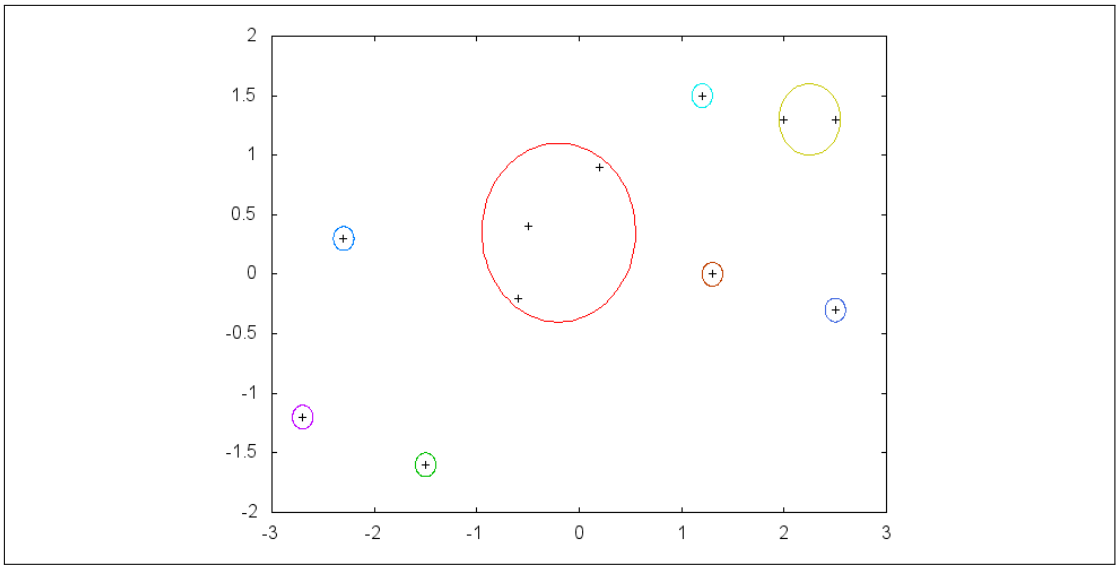


FIGURE 8.7 – Partition intermédiaire de l'algorithme hiérarchique ascendant : 3e regroupement. Deux parties sur 8 ne sont plus unitaires ($K = 8$).

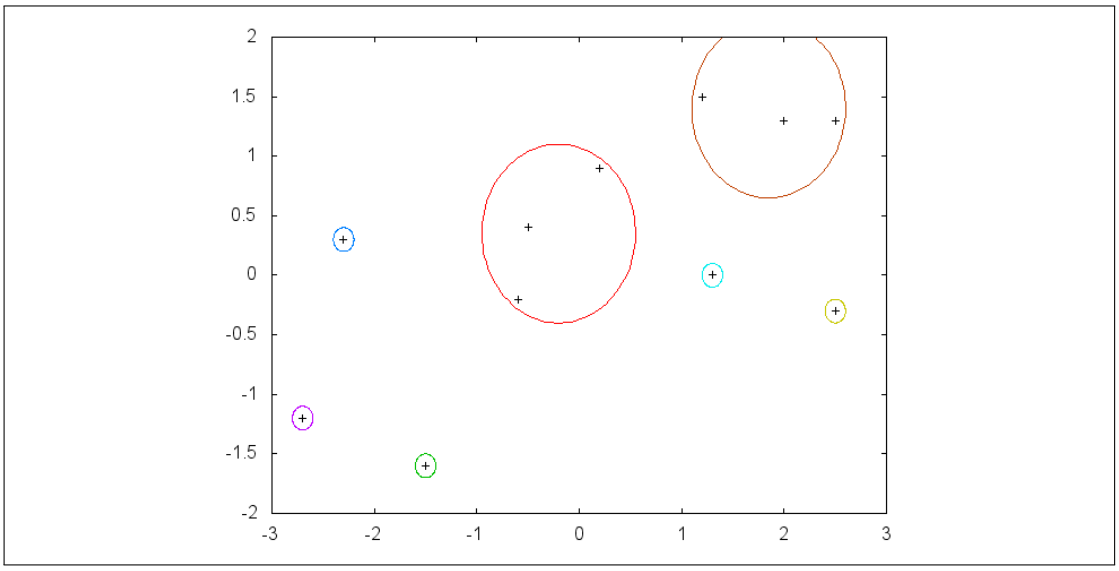


FIGURE 8.8 – Partition intermédiaire de l'algorithme hiérarchique ascendant : 4e regroupement. Deux parties sur 7 ne sont plus unitaires ($K = 7$).

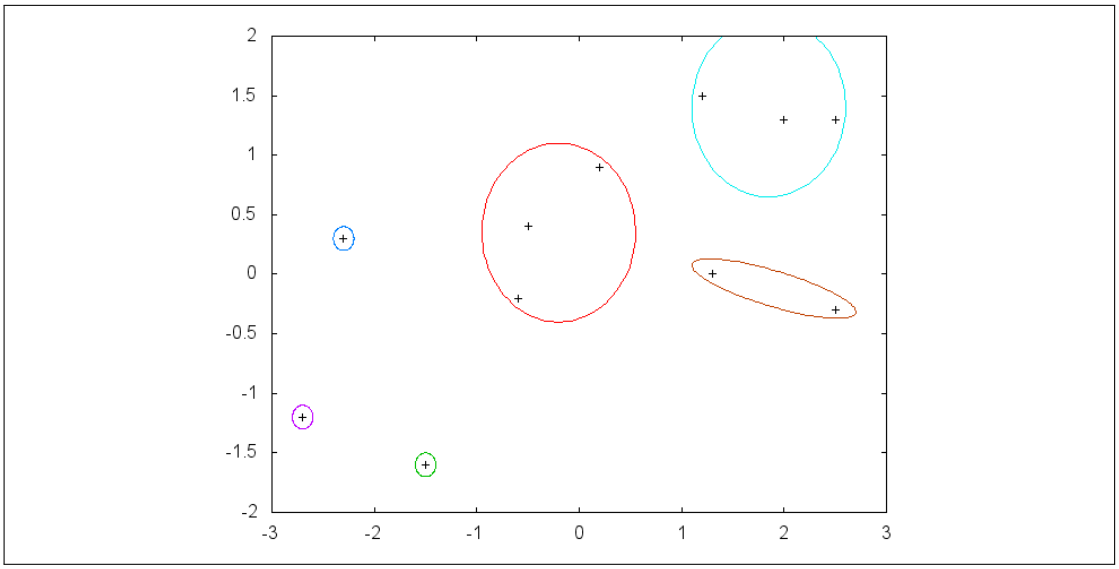


FIGURE 8.9 – Partition intermédiaire de l'algorithme hiérarchique ascendant : 5e regroupement. Trois parties sur 6 ne sont plus unitaires ($K = 6$).

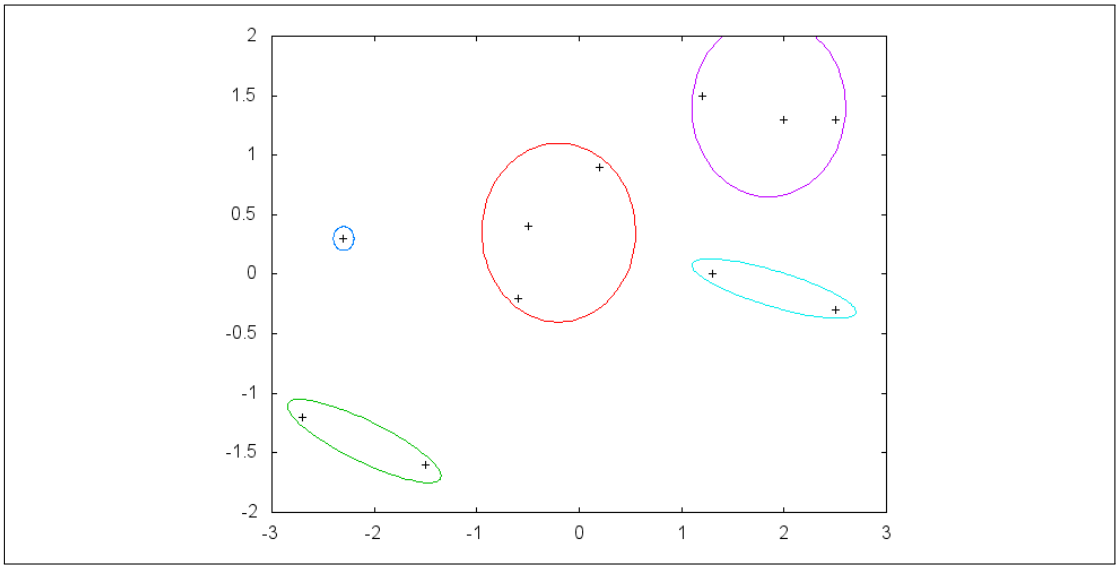


FIGURE 8.10 – Partition finale de l'algorithme hiérarchique ascendant : 6e et dernier regroupement. Quatre parties sur 5 ne sont plus unitaires ($K = 5$).

2. Évaluer la cohérence de chaque partie et scinder la moins cohérente en deux telles que leur cohérence soit maximale. La partition obtenue est alors plus fine.
3. Boucler sur l'étape 2 tant qu'il existe une partie dont la cohérence est inférieure au seuil

L'algorithme 12 page C-10 donne la version en pseudo code de cet apprentissage.

Dans notre exemple, la partition la moins fine de l'espace est décrit par la figure 8.11. L'algorithme part de cette partition pour en former de plus fines jusqu'à atteindre le but escompté. Les figures 8.12, 8.13, 8.14, 8.15 et 8.16 illustrent cette recherche d'une partition admissible.

A la fin de l'algorithme, la partition comporte 6 parties dont 4 non unitaires et 2 unitaires. Nous qualifions de « non classés » les deux éléments de ces parties unitaires.

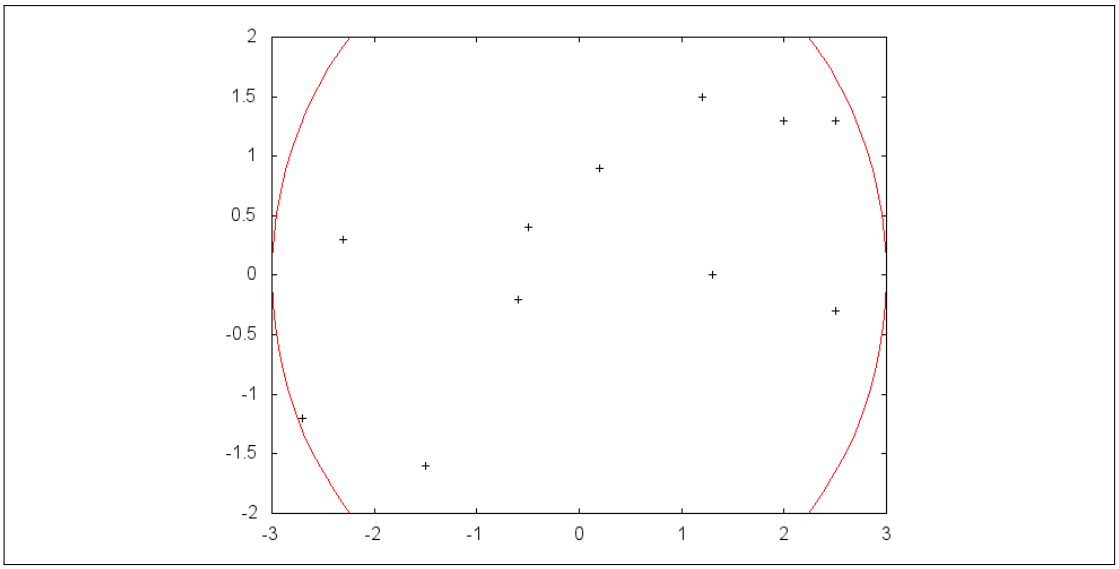


FIGURE 8.11 – Partition initiale de l'algorithme hiérarchique descendant : la moins fine existante sur l'espace d'entrée. Une seule partie regroupe les 11 éléments de l'ensemble d'apprentissage ($K = 1$).

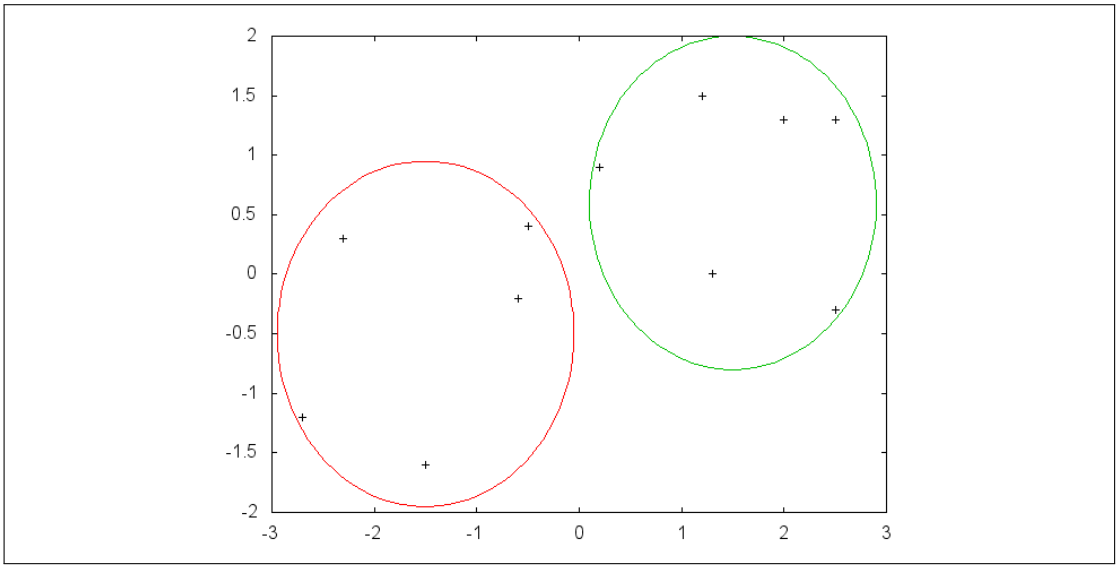


FIGURE 8.12 – Partition intermédiaire de l'algorithme hiérarchique descendant : 1re scission. Deux parties regroupent les 11 éléments de l'ensemble d'apprentissage ($K = 2$).

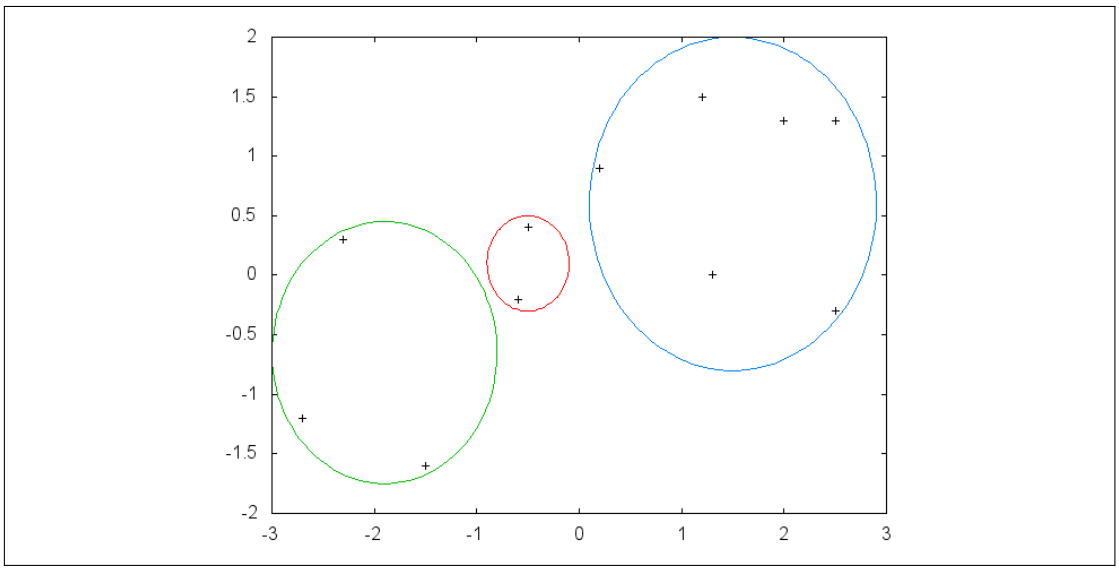


FIGURE 8.13 – Partition intermédiaire de l'algorithme hiérarchique descendant : 2e scission. Trois parties regroupent les 11 éléments de l'ensemble d'apprentissage ($K = 3$).

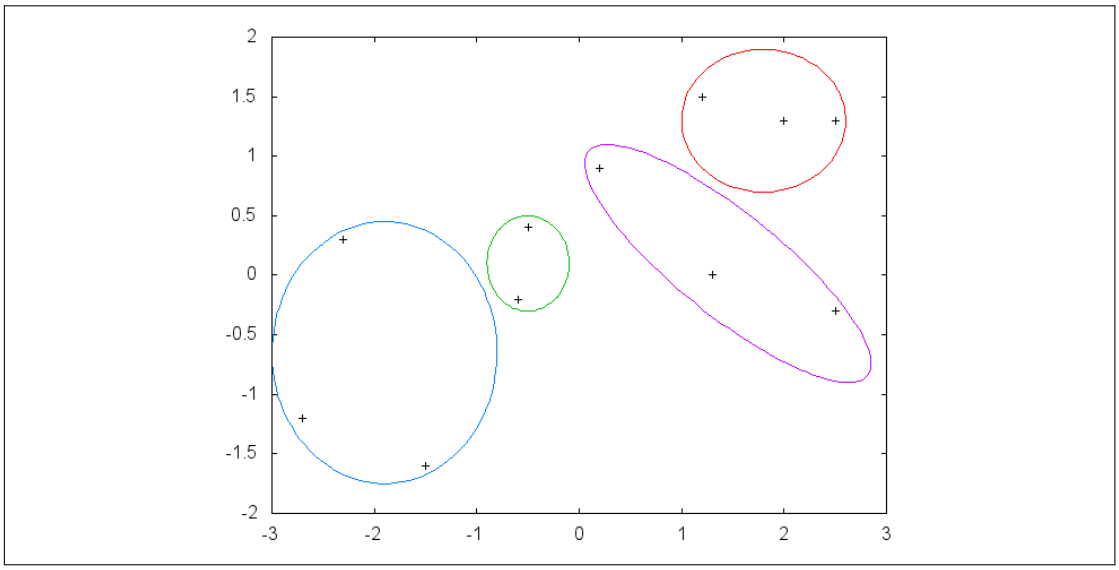


FIGURE 8.14 – Partition intermédiaire de l'algorithme hiérarchique descendant : 3e scission. Quatre parties regroupent les 11 éléments de l'ensemble d'apprentissage ($K = 4$).

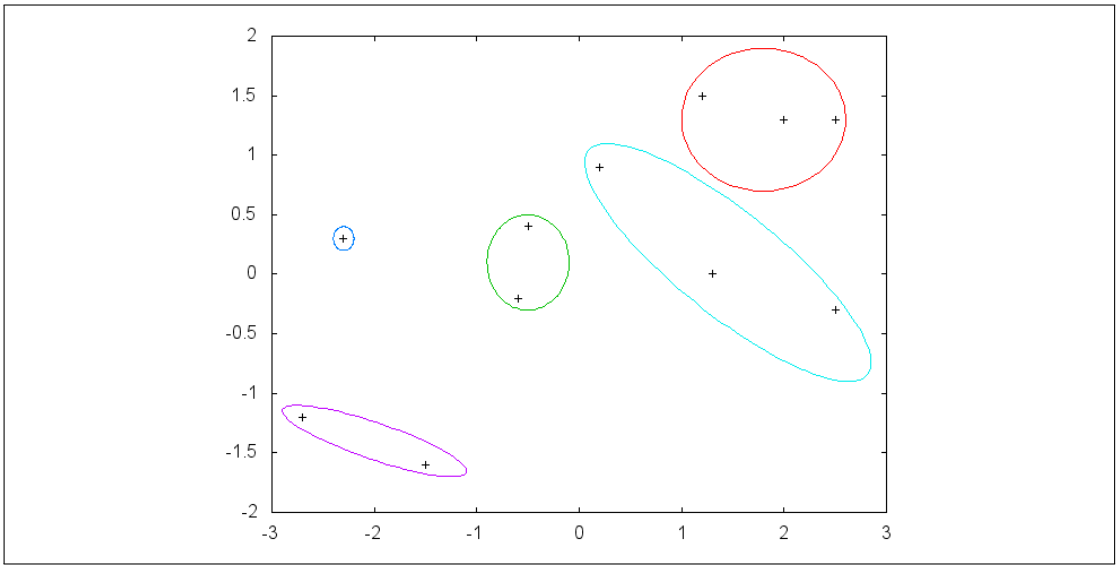


FIGURE 8.15 – Partition intermédiaire de l'algorithme hiérarchique descendant : 4e scission. Cinq parties regroupent les 11 éléments de l'ensemble d'apprentissage dont une partie unitaire ($K = 5$).

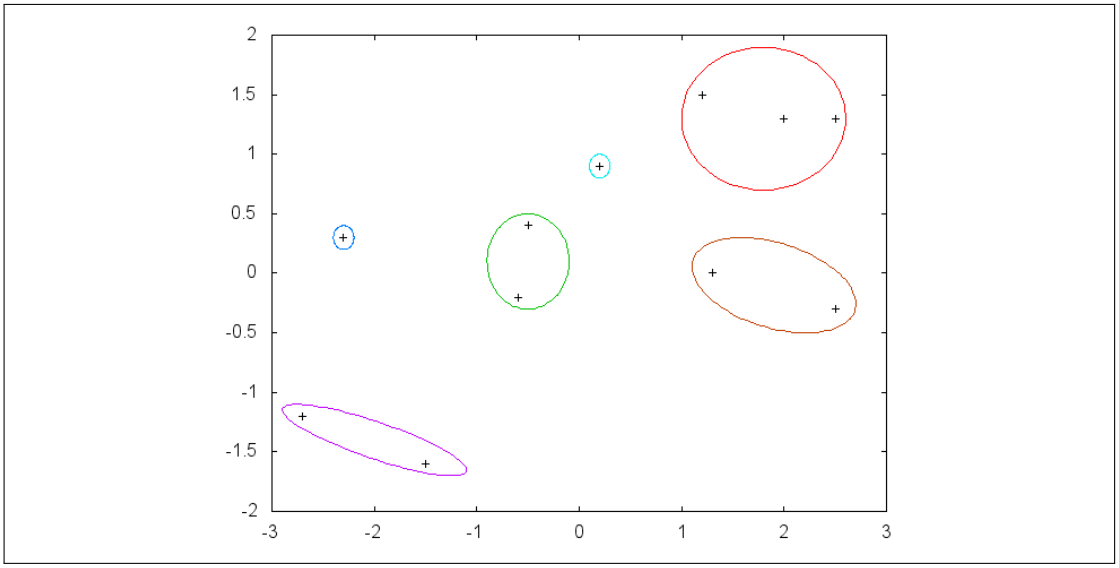


FIGURE 8.16 – Partition finale de l'algorithme hiérarchique descendant : 5e et dernière scission. Six parties regroupent les 11 éléments de l'ensemble d'apprentissage dont deux parties unitaires ($K = 6$).

Troisième partie

Apprentissage de textes courts d'opinions et recherche de produits

Chapitre 9

Présentation informelle de l'entreprise Onyme SARL et des offres concernées par les contributions

Ce premier chapitre de la troisième partie est consacré à la présentation informelle de l'entreprise Onyme SARL dans laquelle j'ai effectué ma thèse et des offres concernées par les contributions de ce manuscrit.

9.1 Présentation générale de l'entreprise Onyme SARL

Les paragraphes suivants présentent brièvement l'entreprise Onyme SARL et son histoire.

9.1.1 Dénomination de l'entreprise et évolutions de son secteur d'activité

Cette thèse a été réalisée dans le cadre de l'entreprise Onyme SARL.

Créée en 2004 par Fabien Degaugue, elle fut ensuite reprise en 2006 par Thibaud Vibes et Antoine Serniclay.

La société fut d'abord une SSII innovante, alliant les activités classiques d'une SSII (développement d'applications notamment de sites web, assistance/maintenance de ces applications, ...) et une activité de recherche et développement dans le traitement automatique des langues (TAL). Cette recherche se concrétise par des prestations ponctuelles (mais pour certaines récurrentes) telles que l'élaboration de rapports d'analyse d'opinions dans des textes courts ou encore la mise au point d'un module d'extraction automatique d'informations depuis des courriers en 2008.

Onyme SARL obtient le statut de Jeune Entreprise Innovante (JEI) en novembre 2007 ainsi que le soutien de OSEO innovation en 2008, soutien renouvelé en 2010, pour sa R&D dans l'analyse du langage naturel.

Fort de cette reconnaissance, à partir de 2010, la société abandonne progressivement ses activités de SSII classique au profit d'un développement accru d'activités centrées sur le traitement automatique des langues. Elle se recentre ainsi principalement sur l'édition de logiciels ayant le traitement de la langue pour valeur ajoutée. Elle devient notamment éditrice du logiciel Onyme Opinions, solution d'analyse en continu de la satisfaction des clients pour le compte d'entreprises.

N'abandonnant pas son métier de prestataire dans le traitement automatique des langues et obtenant l'agrément Crédit Impôts Recherche (CIR) en 2010, la société participe cette même année au développement d'un agent conversationnel spécialisé dans la vente sur internet. Cette activité de prestation devient cependant anecdotique afin de concentrer les efforts de recherche et développement sur l'édition des logiciels, notamment Onyme Opinions.

9.1.2 Situation géographique

L'entreprise est située à Lille sur le site d'Euratechnologies dont elle est « Membre Fondateur ». Ce statut reconnaît l'implication de la société pour le site dès sa création. En effet, elle fut parmi les premières à s'y installer en octobre 2006 avant même son inauguration.

Récent, à la pointe de la technologie, et inauguré en 2009, le site accueille déjà plus d'une centaine d'entreprises de tailles diverses toutes spécialisées dans le domaine des Technologies de l'Information et de la Communication (TIC).

Ce choix d'emplacement permet de mettre en adéquation le dynamisme de l'entreprise avec son environnement et d'offrir à cette dernière la proximité de services et d'entreprises qui lui permettront de continuer son expansion dans le secteur d'activités.

9.1.3 Clients

Onyme SARL compte, parmi ses clients, une large gamme d'entreprises et d'institutions exerçant dans des milieux professionnels différents. Leur particularité commune réside dans leur besoin en terme de traitement automatique des langues.

Parmi eux, nous citerons :

- Plusieurs enseignes du groupe Redcats :
  - Verbaudet
  - Cyrillus
  - Daxon
  - La Redoute
- Plusieurs enseignes du groupe Commerce BtoC, subdivision du groupe 3 Suisses International
  - 3 Suisses
  - Blancheporte
  - Becquet
  - Vitrine Magique
- Excédence
- Banque Accord
- Crédit Agricole
- AKAO
- Archimed
- Compario

9.2 Présentation des offres et prestations

Nous présentons dans cette section les deux offres et prestations proposées par la société et faisant l'objet d'une étude dans le cadre de cette thèse.

9.2.1 Le logiciel Onyme Opinions

Logiciel de prestations ponctuelles (2008-2010)

La première version du logiciel Onyme Opinions fut conçue en 2008 dans le but de permettre des prestations ponctuelles d'élaboration de rapports d'analyse d'opinions exprimées dans des textes courts. Le client fournit les textes à traiter et les compétences en traitement de la langue de la société permettent la génération d'un rapport depuis ces données. Le client n'accède pas à l'application mais seulement au rapport généré grâce à celle-ci.

Les textes traités, appelés verbatims, constituent les données. Il s'agit dans cette première version de réponses à des sondages visant à connaître des opinions de manière générale sur des sociétés ou même des produits en particulier. Ces réponses sont donc souvent courtes (elles ne dépassent pas en moyenne deux ou trois phrases) et sont rédigées en français. Soulignons également la présence possible de fautes d'orthographe.

Le rapport (ou synthèse) généré se présente sous la forme d'une taxonomie des verbatims, organisée selon une hiérarchie de thèmes et d'idées (voir figure 9.1). Pour chaque idée et chaque thème, le rapport présente le pourcentage de verbatims l'ayant exprimé.

Le but de la recherche sur cette application est de concevoir un algorithme capable de découvrir les idées exprimées dans les différents messages. Afin de garantir la fiabilité des éléments transmis au client, la solution intègre par ailleurs un processus de contrôle des idées attribuées par l'algorithme aux verbatims ainsi que la possibilité de les changer si cela est nécessaire. Ce contrôle est assuré par des personnes appelées codificateurs.

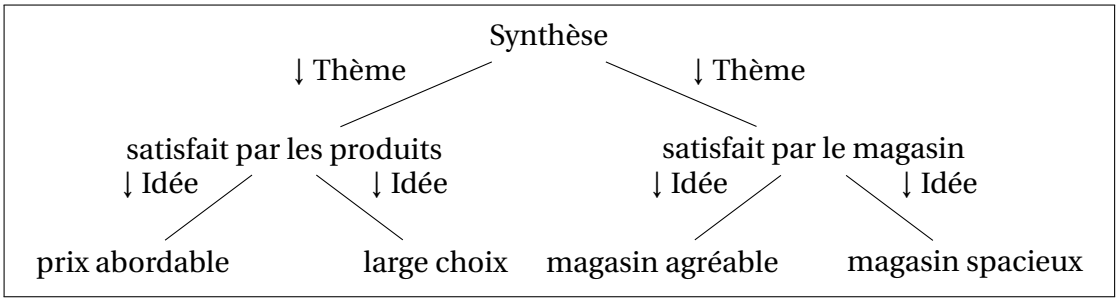


FIGURE 9.1 – Extrait d'une taxonomie utilisée pour classer les verbatims d'opinions. La figure présente deux thèmes évoquant la satisfaction envers les produits et le magasin. Chacun de ces deux thèmes comporte deux idées. Un même verbatim peut évoquer un ou plusieurs de ces thèmes et idées.

A partir de 2010, cette version du logiciel et de l'offre est remplacée progressivement par une nouvelle offre axée davantage sur la satisfaction client. Cependant, quelques traitements sont encore effectués aujourd'hui sur ce modèle.

Logiciel d'analyse de la satisfaction client (2010-)

A partir de 2010, la fonction du logiciel est recentrée sur la problématique plus particulière de l'analyse d'opinions dans le cadre de la satisfaction client. Ce recentrage de l'application introduit une nouvelle dimension dans la prestation : alors qu'auparavant les traitements étaient ponctuels, ils deviennent quotidiens et sur une période atteignant et dépassant l'année. Dès lors, la solution proposée intègre la collecte quotidienne des textes à traiter.

Ces textes (verbatims) sont des réponses aux sondages réalisés auprès des clients des sociétés clientes de l'application en vue de connaître leur opinion sur la société cliente. Ces réponses sont souvent courtes. Le nombre de réponses à traiter est variable en fonction du client mais est réalisé de manière quotidienne pour chacun d'eux. Les réponses sont rédigées en français et saisies par les clients finaux dans un questionnaire disponible en ligne.

Des traitements en dehors de l'analyse de la satisfaction client existent bien que peu fréquents. Cette remarque nous permet de souligner un objectif important : prendre en compte la nécessaire adaptation des algorithmes proposés aux évolutions du marché et surtout aux évolutions de la clientèle. Ces algorithmes doivent être ca-

pables de continuer à fonctionner dans tous les contextes même si une légère perte de qualité peut être tolérée.

Contrairement à la première version, le client accède à l'application disponible en ligne sur laquelle il peut consulter l'analyse des réponses du jour et les comparer avec les réponses de la période antérieure de son choix. Il peut ainsi connaître l'évolution de la satisfaction de ses clients au fil du temps. Les résultats se présentent toujours sous la forme de synthèses des idées présentes dans les verbatims. La différence notable se situe au niveau de la génération dynamique (i.e., à la demande) du rapport comportant la taxonomie des idées et des thèmes. Pour que le client puisse mesurer une évolution dans l'expression des thèmes et des idées, il est primordial que la taxonomie n'évolue que peu au fil du temps.

Le but de la recherche est désormais la conception d'un algorithme capable de reconnaître les idées exprimées dans les verbatims traités au préalable afin de pouvoir les déceler dans les nouveaux verbatims traités. Pour garantir la fiabilité des éléments transmis au client, la solution intègre toujours un processus de contrôle par des codificateurs. Ces contrôles humains, en plus d'assurer la qualité des résultats, fournissent un indicateur journalier potentiel de la qualité du traitement réalisé par l'algorithme.

En complément des réponses textuelles à classer par idées exprimées, le questionnaire peut comporter des questions fermées donnant lieu à des réponses numériques exprimant une note souvent comprise entre 0 pour exprimer un mécontentement important et 10 pour exprimer une satisfaction totale. Ces questions ne sont bien sûr pas à traiter par un algorithme d'analyse d'opinions.

Nous étudions particulièrement un type de questionnaire fondé sur trois questions et dénoté questionnaire NPS (REICHHELD 2003). Ces types de questionnaire se composent des trois questions suivantes :

1. Recommanderiez-vous notre société à vos amis, vos proches, vos collègues ?
2. Pourquoi cette note ?
3. Que pouvons nous faire pour mieux vous satisfaire la prochaine fois ?

La première est une question de type numérique comme celle que nous avons décrite auparavant tandis que les deux dernières sont des questions textuelles sur

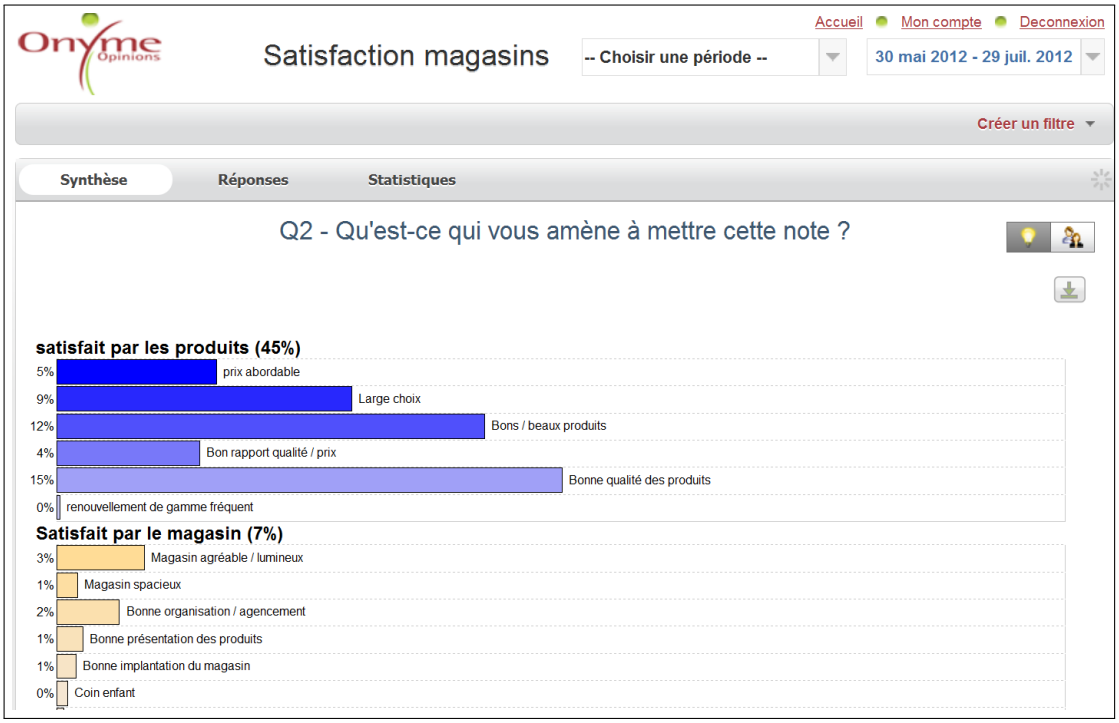


FIGURE 9.2 – Exemple de synthèse d'une question 2 d'un questionnaire NPS. La synthèse a été générée avec les données du 30 mai au 29 juillet 2012. La section de la synthèse montrée ici présente deux thèmes de respectivement 5 et 6 idées.

lesquelles il nous faut reconnaître les idées exprimées. Nous les dénoterons respectivement par question 2 et 3 d'un questionnaire NPS.

Les figures 9.2 et 9.3 donnent des exemples de synthèses sur des questions 2 et 3 d'un questionnaire NPS. Les figures 9.4 et 9.5 donnent des exemples de données sur des questions 2 et 3 d'un questionnaire NPS.

Nous observons au travers des exemples de synthèses quelques exemples d'idées à reconnaître dans les verbatims exprimés par les clients. Nous voyons également comment le regroupement de ces idées par thèmes est exploité afin de rendre la lecture plus claire et d'offrir un indicateur global de représentativité des idées du thème.

Ainsi, dans l'exemple 9.2, on observe les deux thèmes regroupant les idées de satisfaction concernant les produits et le magasin. Pour le premier thème, les idées vont du prix de ces produits jugés comme abordable à la qualité globale des produits. Pour le second thème, les idées vont du magasin jugé comme agréable et lumineux à son implantation jugée judicieuse.

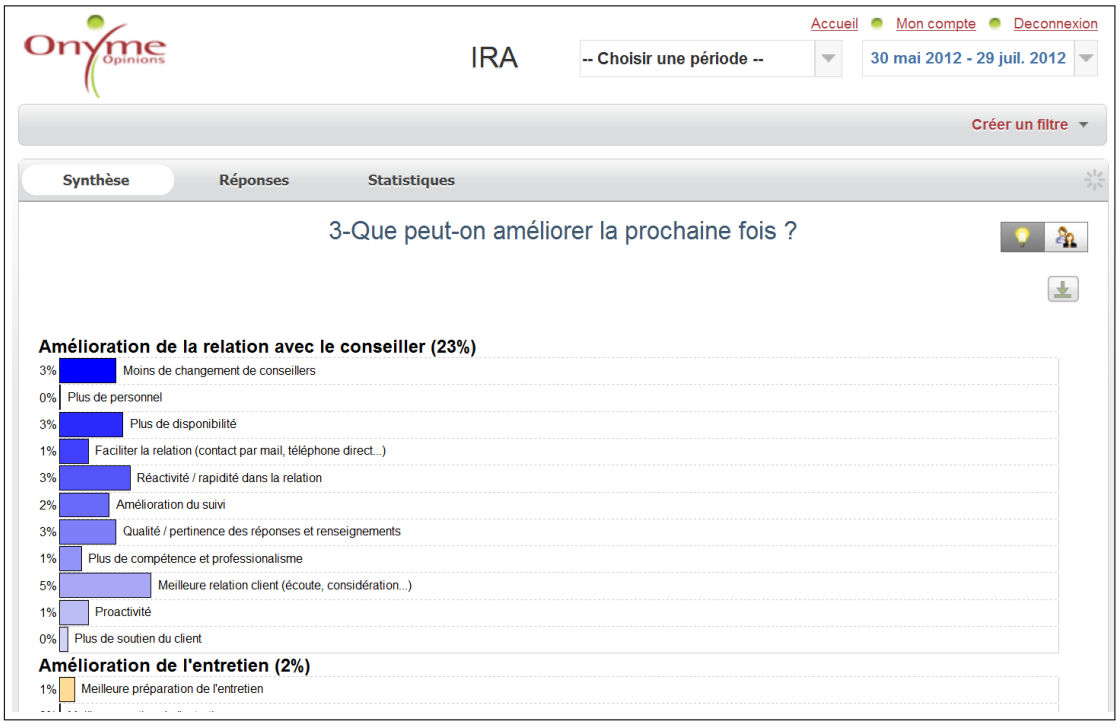


FIGURE 9.3 – Exemple de synthèse d'une question 3 d'un questionnaire NPS. La synthèse a été générée avec les données du 30 mai au 29 juillet 2012. La section de la synthèse montrée ici présente un thème de 11 idées.

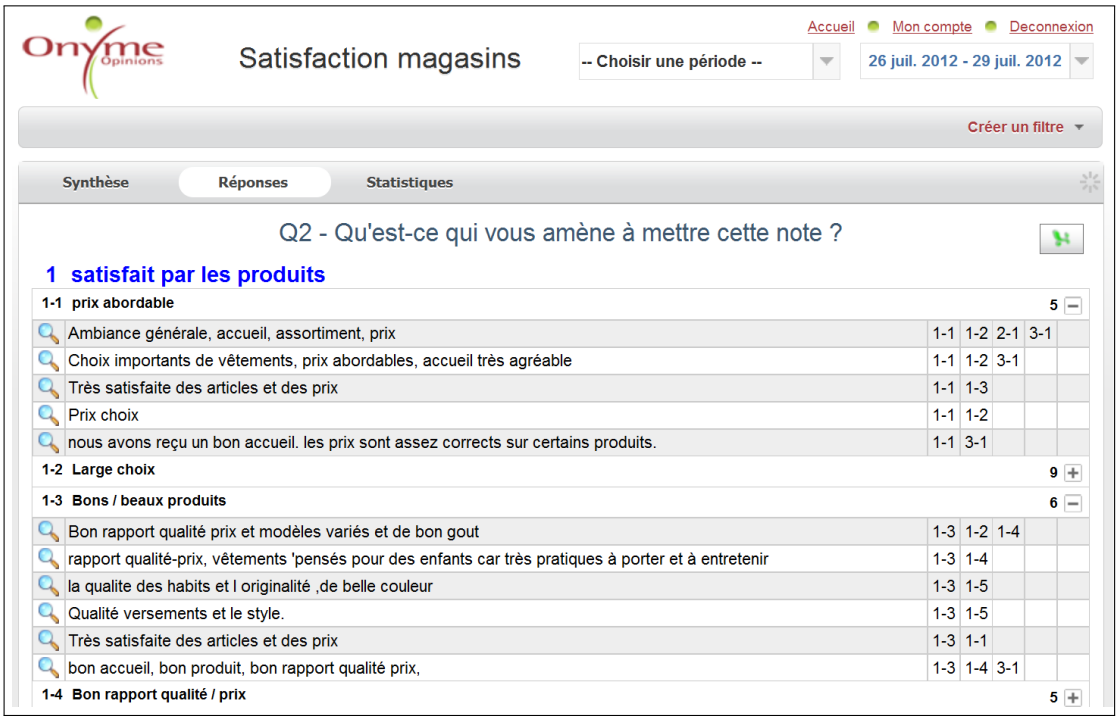


FIGURE 9.4 – Exemple de données d'une question 2 d'un questionnaire NPS. La synthèse a été générée avec les données du 26 juillet au 29 juillet 2012. La section de la synthèse montrée ici présente les verbatims de deux idées du même thème. Ces idées comportent respectivement 5 et 6 verbatims dont un verbatim commun aux deux idées (verbatim comportant les deux idées).

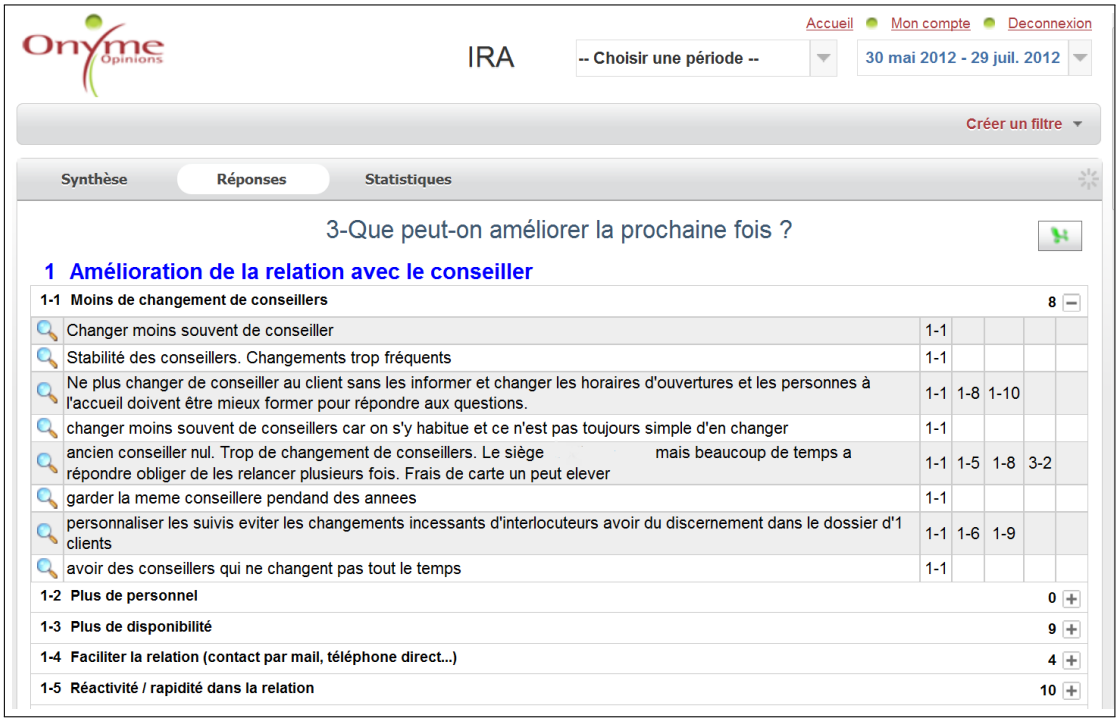


FIGURE 9.5 – Exemple de données d'une question 3 d'un questionnaire NPS. La synthèse a été générée avec les données du 30 mai au 29 juillet 2012. La section de la synthèse montrée ici présente les 8 verbatims d'une idée.

En étudiant quelques verbatims du premier thème présentés à la figure 9.4, nous pouvons observer que beaucoup de ceux-ci expriment plusieurs idées comme par exemple le verbatim « Très satisfaite des articles et des prix » qui regroupe les idées de « prix abordable » et « bons/beaux produits », tous deux dans ce thème. De manière globale, sur les dix verbatims présentés, aucun ne comporte qu'une seule idée, six en comportent deux, trois en comportent trois et un seul en comporte quatre.

Dans l'exemple 9.3, le thème regroupant les idées d'améliorations de la relation avec le conseiller est présenté. Il regroupe des idées telles que la volonté de changer moins souvent de conseiller.

En étudiant quelques verbatims exprimant cette idée (figure 9.5), nous constatons cette fois que peu d'entre eux expriment plusieurs idées. Sur les huit verbatims présentés, cinq ne comportent qu'une seule idée (la volonté de changer moins souvent de conseiller), deux en comportent trois et un en comporte quatre.

Soulignons également dans ces quelques exemples quelques-unes des fautes d'orthographe et de syntaxe commises :

1. Qualité **versements** et le style.
2. Le siège **mais** beaucoup de temps **a** répondre **obliger** de les relancer plusieurs fois.
3. Bon rapport qualité prix et modèles variés et de bon **gout**
4. garder la **meme conseillere pendand** des **annees**

Les fautes relevées ici sont de différentes natures :

1. des absences d'accentuation : meme, conseillere, annees, gout, a
2. des remplacements de mots par d'autres mots existants proches : versements (vêtements), mais (met)
3. une transition syntaxique sans marquage : a répondre obliger de

La saisie en ligne des réponses par les clients finaux des entreprises clientes de l'application conduit ainsi à un « bruit » important dans les textes à analyser.

9.2.2 Prestation de recherche : développement d'un agent conversationnel spécialisé dans la vente sur internet

Durant l'année 2010, la société a participé en tant que prestataire de recherche à un projet concernant l'élaboration d'un agent conversationnel spécialisé dans la vente sur Internet. Le but de ce projet ambitieux, réalisé par plusieurs sociétés, était de concevoir un vendeur virtuel capable de gérer de manière automatique l'ensemble des aspects de la relation de vente entre le vendeur et les clients.

Afin de réduire le délai de production de la première version, les ambitions ont été réduites aux deux fonctions primaires du vendeur, à savoir les interactions sociales avec le client et l'assistance dans la recherche d'un produit correspondant à ses attentes.

L'outil se présente comme un cadre intégrable sur le site web du e-commerçant invitant l'internaute à dialoguer avec lui par l'intermédiaire d'une succession de textes courts saisis dans une boîte de dialogue. L'agent vendeur répond alors à l'internaute en l'incitant par exemple à préciser sa recherche de produit. La figure 9.6 montre le début d'une telle conversation.

Le rôle d'Onyme SARL a été défini sur deux axes de recherche sur les textes produits par les internautes :

1. La reconnaissance du type d'interaction sociale afin de pouvoir y répondre ;
2. La traduction de la recherche d'un produit selon une liste de critères exprimée en langage naturel en une requête (un filtre) pertinente et compréhensible par un système de gestion de base de données.

Les exemples 9.1 et 9.2 donnent des exemples respectifs de ces deux types de textes produits par les internautes. Soulignons également que pour simplifier le problème, nous supposons que les internautes ne recherchent des produits qu'au travers de leurs caractéristiques propres et non au travers de caractéristiques d'autres produits ou plus généralement de caractéristiques extérieures. Ainsi, le traitement des textes tels que ceux présentés dans l'exemple 9.3 ne figurent pas dans nos objectifs. D'une manière générale, satisfaire ces recherches de produits nécessiterait de disposer de compétences/connaissances spécifiques sur les domaines concernés. Dans

FIGURE 9.6 – Début d'une conversation avec Fred, le vendeur virtuel. La recherche initiale de l'internaute concerne les canapés. Cette recherche est assez vague. L'agent obtient le renseignement complémentaire que ce canapé doit être de préférence en cuir par l'intermédiaire de la conversation avec l'internaute.

nos exemples, nous devrions avoir des connaissances sur la mode et la mécanique des vélos BMX. Celles-ci relèvent de vendeurs experts.

Exemple 9.1

Quelques exemples de textes à caractère social que nous devons reconnaître :

- *Bonjour*
- *Comment ça va ?*
- *Merci*
- *Au revoir*

Exemple 9.2

Quelques exemples de textes de recherche de produits que nous devons reconnaître :

- *As tu des canapés marron ?*
- *Je voudrais une lampe pour moins de 20 euros*

– *Peux tu me donner un bureau en chêne et une lampe bleue pour aller avec.*

Soulignons l'absence de traits d'unions dans le premier et troisième texte ainsi que le signe de ponctuation erroné en fin du troisième texte.

Exemple 9.3

*Quelques **contre**-exemples de textes de recherche de produits que nous ne devons pas traiter :*

– *As tu une chemise tendance assortie avec un pantalon bleu ?*

– *Je voudrais une nouvelle chaîne pour mon vélo BMX modèle xx.*

Soulignons l'absence du trait d'union dans le premier texte.

Dans la pratique, les deux types d'interactions mentionnées auparavant ne sont pas toujours dissociées. Bien souvent, l'internaute choisit de combiner les deux types d'interactions dans un seul et même texte transmis à l'agent conversationnel comme c'est le cas dans la conversation donnée en exemple à la figure 9.6. L'interaction sociale de remerciement est alors initiée par l'internaute dans le même texte que sa recherche portant sur les articles de type canapé.

Notons également que notre travail ne porte pas sur la génération de la conversation aussi bien en terme de génération du langage que dans les choix d'orientation de la conversation (par exemple le choix de continuer la conversation de recherche de produit par la précision de la matière du canapé). Cette partie du traitement est gérée par un autre acteur du projet et notre travail se limite donc aux deux axes évoqués auparavant.

La correction orthographique des textes est également assurée par ailleurs par un autre acteur du projet. Nous considérons donc les textes que nous traitons comme orthographiquement corrects. La correction proposée n'étant cependant pas syntaxico-sémantique, nous devons prendre en compte l'absence de rigueur possible à un niveau syntaxique comme cela est souligné dans nos exemples.

Nous ne décrivons dans la suite de cette thèse que le travail réalisé sur l'axe dédié à la recherche de produits en raison d'une contribution plus importante sur celui-ci que sur celui dédié aux interactions sociales.

9.3 Présentation générale des contributions

Dans cette section, nous introduisons de manière globale l'ensemble des contributions de cette thèse.

Dû à leur contexte industriel, nos travaux ont fait l'objet de la réalisation de plusieurs programmes, écrits en langage Java, dont quelques métriques clés sont donnés en annexe A. La plupart de ces programmes sont d'ores et déjà exploités pour des traitements de données clients.

9.3.1 Représentation vectorielle de textes courts d'opinions pour l'apprentissage

Notre première contribution porte sur l'offre Onyme Opinions et l'enjeu de classification des verbatims selon l'opinion dégagée.

Cette thèse a débutée en 2009. Comme nous l'avons souligné dans la présentation de l'offre, le besoin industriel lié à cette contribution a évolué durant notre étude pour passer d'un besoin d'analyse ponctuelle récurrent à un besoin d'analyse journalier en continu. Cela a rendu nécessaire l'adaptation de nos solutions aux changements.

Les méthodes d'apprentissage que nous avons présentées dans la partie II permettent l'adaptation aux changements mais requièrent pour fournir un résultat pertinent un espace d'entrée adapté. Les représentations vectorielles sont particulièrement pertinentes car elles offrent un espace d'entrée exploitable par des méthodes d'apprentissage éprouvées.

Notre contribution dans ce domaine porte ainsi de manière générale sur une étude des méthodes permettant la représentation de textes en vue de cet apprentissage. Nous avons étudié l'impact de différents choix faits en terme de représentation sur les performances obtenues en apprentissage.

Entre autres, la saisie des textes via un formulaire en ligne engendre très souvent des textes fortement bruités à cause de l'orthographe. Nous proposons d'employer

des représentations basées sur les caractères et les phonèmes pour les mots dans le but d'obtenir une distance hybride entre mots.

Les évolutions de l'offre nous ont conduit à considérer aussi bien des apprentissages non supervisés, utiles pour des traitements ponctuels, que supervisés, utiles pour des traitements continus.

9.3.2 Recherche de produits dans une base de données à partir de requêtes en texte libre

Cette contribution porte sur le projet de l'agent conversationnel vendeur sur Internet.

Le problème majeur auquel nous avons à faire face dans la recherche des produits est celui d'établir un lien entre les caractéristiques énoncées par l'internaute dans sa recherche et les caractéristiques se trouvant dans le catalogue de produits du site de e-commerce.

Nous montrons comment mettre en relation les lexiques employés par les internautes et les lexiques employés par la base de données. Si cela est suffisant pour gérer une recherche monocritère telle que « Je cherche un canapé », cela ne l'est pas à partir de l'instant où la recherche devient multicritères telle que « Je cherche un canapé en cuir ». La requête contient en effet deux caractéristiques de l'article recherché : le type canapé et la matière cuir.

Plus généralement, nous montrons dans cette partie comment gérer des requêtes plus complexes mettant en jeu plusieurs produits telles que par exemple « Je cherche un canapé en cuir pour moins de 5000 euros et un fauteuil pas trop cher ».

Soulignons également que le dialogue avec l'agent par des textes saisis sur Internet nous oblige à tenir compte de difficultés liées à un non-respect fréquent de l'orthographe et de la syntaxe et ce, même si une correction orthographique est proposée en amont de nos traitements.

9.3.3 Gestion des couleurs dans les bases de données

Cette contribution est complémentaire au projet de l'agent conversationnel vendeur sur Internet même si elle porte en réalité sur une problématique plus large qu'est la gestion des couleurs dans une base de données.

Nous montrons comment représenter les couleurs dans une base de données afin d'établir des recherches efficaces et pertinentes sur les couleurs disponibles.

Dans le cadre de la vente, cette contribution peut être employée afin de rechercher dans la base de produits, ceux correspondant à la couleur recherchée par l'internaute. Ainsi, la requête « Je cherche un canapé marron » déjà mentionnée pourra être satisfaite par la fourniture d'un canapé chocolat.

9.4 Conclusion

Nous avons présenté les différentes offres et prestations faisant l'objet d'une contribution dans cette thèse. Nous avons également présenté brièvement les différentes contributions que nous détaillons dans les chapitres suivants de cette partie de la thèse.

Chapitre 10

Représentation vectorielle de textes courts d'opinions pour l'apprentissage

La première section de ce chapitre a fait l'objet d'une publication à la conférence Rencontres des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL), édition 2010, Montréal, Canada (TROUVILLIEZ 2010). Dans celle-ci, nous présentons nos premières démarches dans l'étude des techniques de représentations de textes courts d'opinions et leur impact sur un apprentissage non supervisé réalisé dans le cadre de traitements ponctuels.

Dans la deuxième section, nous nous focalisons sur la reconnaissance des verbatims comportant plusieurs idées. Nous exposons notre stratégie pour prendre en compte cette reconnaissance lors de l'apprentissage non supervisé.

Dans la troisième section, nous présentons notre stratégie de correction orthographique hybride basée sur une distance entre les phonèmes et les caractères des mots erronés et ceux corrects.

Dans la quatrième et dernière section, nous expliquons en quoi l'évolution de l'offre nous a conduit à introduire de nouvelles données à prendre en compte pour la représentation des textes et l'impact de celles-ci sur l'apprentissage devenu supervisé, réalisé à partir d'un échantillon des traitements antérieurs. Nous évaluons enfin les fonctions de classification ainsi apprises à l'aide d'échantillons de test.

10.1 Étude préliminaire de la représentation dans le cadre de traitements ponctuels en apprentissage non supervisé

Avec le développement d'Internet et plus généralement des moyens de communication, le volume de données d'informations à traiter ne cesse d'augmenter. Il est maintenant nécessaire d'employer des méthodes d'analyse automatique.

Dans cette section, nous présentons une approche de fouille d'opinions à partir de textes courts et expliquons en quoi notre choix d'utilisation de regroupements autour des idées exprimées nous a conduit à opter pour une représentation vectorielle. Nous présentons également les différents traitements intégrés à notre chaîne de traitement (traitement de la négation, lemmatisation, racinisation, synonymie ou même polysémie des mots) et discutons leur impact sur la qualité des regroupements obtenus.

10.1.1 Introduction de l'étude préliminaire

On observe la croissance rapide de sources d'informations reflétant des opinions sur des sujets variés : films, actualités, valeur d'une entreprise, prestations d'un commerçant, prestations d'un site Internet, ... Ces sources prennent des formes diverses telles que des forums, des blogs, des sondages en ligne, des sondages par téléphone, ... Les textes sont assez courts et dépassent rarement 4 phrases. Dans ce contexte, plusieurs stratégies de fouilles d'opinions ont vu le jour. Alors que certains travaux visent à une analyse de sentiments afin de déterminer l'opinion des auteurs sur le sujet en question (POIRIER et al. 2008), nous nous intéressons à la problématique du regroupement afin d'extraire les principales idées développées. Nous faisons appel à deux techniques différentes : des méthodes d'analyse extrayant l'information des textes dans une représentation et des méthodes de regroupements non supervisés transformant cette information en classes d'opinions (Fig. 10.1).

10.1.2 Représentation et traitements

Généralités, constats et problèmes

Nous commençons par décrire la tâche et faisons quelques constats quant aux problèmes rencontrés. La tâche consiste à effectuer des regroupements de textes selon l'opinion qu'ils expriment. Ces textes sont la plupart du temps issus de sondages d'opinions sur un produit, un achat, une prestation, un entretien,...

Nous souhaitons utiliser des techniques d'apprentissage non supervisé pour réaliser ces regroupements et ce principalement en raison de l'adaptation de ces techniques aux changements dans les données à regrouper. Nous espérons ainsi obtenir un système robuste aux changements de clients, de données à traiter,... Cette contrainte est bien sûr essentielle dans un contexte industriel.

Premièrement, les textes que nous avons à traiter ont une taille moyenne de deux phrases excédant rarement 5 mots chacune : la taille des représentations de ces textes est alors petite. Ils reflètent des opinions d'auteurs différents, n'utilisant pas le même registre de langue et n'exprimant que peu d'idées / opinions, rarement répétées dans un même texte.

Pourtant, nous ne pouvons considérer justement ces textes comme porteurs d'une seule idée. Le nombre d'idées véhiculés par chacun d'eux dépend de la finesse de la synthèse des idées que nous souhaitons obtenir (Ex. : « plus de choix sur les chaussures et les pulls » comporte l'unique idée « plus de choix » ou les deux idées « plus de choix de chaussures » et « plus de choix de pulls » selon la précision des idées de la synthèse). En moyenne, pour les synthèses que nous souhaitons générer, entre 40 et 70% des textes comportent plusieurs idées. Cela constitue une difficulté pour les algorithmes d'apprentissage qui ne sont conçus la plupart du temps que pour établir des partitions de l'espace d'entrée. Afin de simplifier notre étude préliminaire, nous considérons dans un premier temps uniquement des synthèses d'idées conduisant à établir une partition de l'espace d'entrée. Nous verrons dans un second temps nos stratégies pour gérer plus finement ce problème.

Troisièmement, les auteurs des textes n'écrivent pas toujours dans un langage très rigoureux. il n'est pas rare de trouver des fautes d'orthographe ou des abrévia-

tions. Il est dans ce cas important que les représentations utilisées possèdent une certaine robustesse face à une orthographe difficile.

Quatrièmement, le rapport qualité / temps obtenu est un facteur important de la qualité du système en contexte industriel. Le temps de traitement d'un verbatim ne devrait pas excéder 5 secondes dans le pire des cas afin que les traitements puissent s'effectuer dans un délai raisonnable. L'estimation de l'évolution de la volumétrie des textes à traiter vers la hausse nous conduit par ailleurs à estimer en baisse cette limite maximale.

Les représentations

Nous pensons qu'une représentation vectorielle des textes constitue un choix judicieux afin d'obtenir un rapport qualité / temps convenable pour nos regroupements. Nous les préférons donc à des représentations plus complexes et donc plus coûteuses en terme de temps CPU telles que des représentations sémantiques.

L'utilisation de tous les mots des textes engendre une taille de représentation vectorielle excessive, pouvant être difficile à traiter aussi bien en terme de temps CPU qu'en terme de coût mémoire. Il est courant d'effectuer des traitements visant à la simplifier en regroupant ou supprimant certaines composantes des vecteurs. Cette simplification des descripteurs a un impact sur la qualité finale du traitement : si les textes ne sont pas correctement représentés, il est peu vraisemblable que le résultat obtenu soit probant. Certains chercheurs préconisent de ne pas systématiquement employer de méthodes de simplification mais au contraire de réfléchir à leur pertinence et au fait que des mots supprimés des représentations peuvent être primordiaux pour des traitements proches de la sémantique (RILOFF 1995).

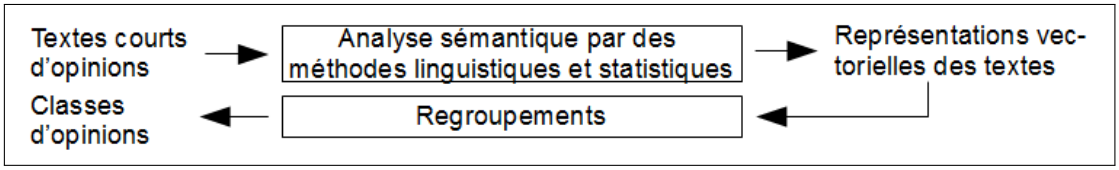


FIGURE 10.1 – Processus de traitement des textes d'opinions par regroupements. Le schéma présente les deux phases constituant le traitement et les données en entrées et sorties de ces phases.

Lemmatisation / racinisation

Ces méthodes identifient les mots différents à cause des règles syntaxiques et grammaticales mais ayant une sémantique identique. La lemmatisation identifie la fonction grammaticale et le lemme du mot avec l'aide éventuelle du contexte, solution étudiée ici. La racinisation ne résout jamais le contexte mais identifie selon la forme et la langue utilisée, la fonction grammaticale du mot et en déduit sa racine. La lemmatisation simplifie les représentations de mots dont le radical varie mais pas celles des mots occupant une fonction grammaticale différente. La racinisation, au contraire, est capable d'effectuer des rapprochements entre les mots occupant une fonction grammaticale différente mais pas entre ceux dont le radical varie et provoque de plus un rapprochement non désiré entre les mots sémantiquement différents mais ayant un radical commun.

Nous rappelons que nous notons $lemmatisation(mot)$ et $racinisation(mot)$, les fonctions donnant les lemmes et la racine d'un mot.

Nos choix se sont portés sur *TreeTagger* (SCHMID 1994), étiqueteur multilingue réalisé par le laboratoire de l'Université de Stuttgart, pour la lemmatisation et sur *Snowball*, sous projet de Apache Lucène [1], pour la racinisation. Ils ont été testés sur différents jeux de tests. Ils ne commettent pratiquement aucune erreur en contexte orthographique correct. *TreeTagger* semble même parvenir à désambiguïser correctement des phrases comportant des homonymes comme le verbe « porter » et le nom « porte ». Sur un corpus mal orthographié, les résultats de lemmatisation s'effondrent. Aucun lemme correct n'a été identifié sur les termes inconnus et peu de fonctions grammaticales l'ont été. Du côté de la racinisation, les résultats sur un corpus mal orthographié sont mitigés. Les mots pour lesquels la faute porte sur la partie considérée comme la racine n'ont pas été corrigés. De même, certaines fautes portant sur le suffixe ont modifié la racine retournée par *Snowball* (exemple : « jouet » donne la racine « jouet » alors que « jouer » donne la racine « jou »).

Ces tests sur la lemmatisation et la racinisation montrent que la lemmatisation est peu efficace dans un contexte orthographique difficile. Cela est surtout un handicap si le terme affecté est un terme décisif pour les regroupements. La racinisation se

1. http ://lucene.apache.org/

révèle être une solution si la faute ne modifie pas la racine du mot. Ces constats nous ont conduits à supposer que la combinaison des deux méthodes permettrait de tirer parti des avantages de chacune des deux stratégies : la lemmatisation permet de regrouper les mots dont le radical évolue et la racinisation, les différentes formes grammaticales d'un même concept tout en corrigeant éventuellement quelques fautes d'orthographe. Les résultats intermédiaires obtenus après la lemmatisation sont conservés et utilisés lors des rapprochements sémantiques. Il serait également possible d'utiliser les statistiques ainsi obtenues pour déterminer si deux mots proches après racinisation le sont à cause de leur sémantique.

De plus, effectuer une lemmatisation en amont de la racinisation permet une meilleur gestion des mots dont la découverte du radical est ambiguë.

Exemple 10.1

Dans le cas des mots « satisfait » et « satisfaite », la racinisation donne le résultat :
$racinisation(satisfait) = satisf$
$racinisation(satisfaite) = satisfait$
car « satisfait » est reconnu à tort comme le verbe « satisfer » à l'imparfait et « satisfaite » comme un adjectif. Les deux racines sont alors différentes de part l'erreur sur la racinisation de « satisfait ».

En passant par une lemmatisation préalable, la racinisation donne le résultat :
$racinisation(lemmatisation(satisfait)) = racinisation(satisfaire) = satisfai$
$racinisation(lemmatisation(satisfaite)) = racinisation(satisfaire) = satisfai$

Les deux racines trouvées sont alors identiques.

Pondération des mots selon leurs sens

La pondération vise à accorder plus d'importance à certains attributs, à en pénaliser d'autres en allant jusqu'à les retirer (pondération nulle) et s'oppose à l'équi-importance qui considère que tous les mots d'un message apportent la même quantité d'informations. Utiliser une pondération est particulièrement intéressant dans un but de regroupements pour distinguer les mots véhiculant beaucoup d'informa-

tions, de ceux n'en apportant que peu. Deux types d'approches existent. Les premières, méthodes statistiques, se basent sur les occurrences des mots dans le corpus pour déterminer l'importance de ceux-ci en contexte. Les formules du TF.IDF (SPARCK JONES 1972) et de l'entropie (SHANNON 1948) sont des exemples de mesures d'importance utilisées dans de telles approches. Dans le cadre de regroupements, un mot qui apparaît dans la majorité des textes n'est pas assez discriminant et conduit à en rapprocher trop. Un mot apparaissant très rarement conduit à en rapprocher trop peu sur des idées peu représentatives. Les deuxièmes, méthodes linguistiques, consistent à établir une liste des mots à éliminer, considérée comme peu évolutive et de taille raisonnable (SALTON, WONG et YANG 1975) (SALEM 1987). Le contenu et la longueur de cette liste noire sont différents en fonction de la nature du traitement. Quelle que soit l'approche retenue, la suppression à ce stade d'un descripteur important pour le corpus est définitive et provoque une dégradation sensible des résultats.

Dans notre contexte, les opinions étant rarement répétées, il est important de faire attention à ne pas supprimer de mots utiles pour éviter les pertes d'informations. Utiliser une liste noire pour supprimer les mots inutiles permet le contrôle des mots retirés. Considérer les autres mots comme équi-importants ne semble pas être une solution efficace car certains véhiculent plus le sens global des textes que les autres. Utiliser une technique statistique de pondération semble donc judicieux. La formule du TF.IDF n'est pas appropriée car sa pertinence repose sur l'hypothèse de répétition des sens importants dans un texte, non vérifiée si les textes sont très courts. De même, l'IDF seul privilégie les mots très rares. Utile dans un cadre de recherche de termes discriminants, cela a pour effet dans notre cas de provoquer le rapprochement de mots issus du niveau de langage du répondant, plus que sur les idées exprimées. On peut en revanche supposer que l'importance des mots pour les textes est équivalente à l'importance des mots pour l'ensemble des textes. Nous pouvons utiliser simplement l'entropie de Shannon appliquée sur l'ensemble des documents pour que les mots ayant une probabilité d'apparition moyenne dans l'ensemble du corpus soient privilégiés.

Soient I_{mot}, P_{mot} et Occ_{mot} représentant respectivement l'importance, la probabilité d'apparition et l'occurrence du mot dans le corpus, et Nb_{mots}, le nombre de mots au total dans le corpus. La formule 10.1 donne les relations entre ces variables.

$$\begin{aligned} P_{mot} &= \frac{Occ_{mot}}{Nb_{mots}} \\ I_{mot} &= -P_{mot} * log(P_{mot}) \end{aligned} \tag{10.1}$$

Deux mots sémantiquement proches devraient obtenir des notes qui reflètent leurs liens. Pour assurer cela, il est courant d'associer à la représentation vectorielle des méthodes permettant d'identifier les liens sémantiques entre les mots du corpus. Nous proposons d'utiliser une méthode linguistique basée sur une ressource lexicale. La plus connue est le *Wordnet* de Princeton (MILLER 1995), créé pour la langue anglaise et fondateur des « *Wordnets* » : ressources aux caractéristiques similaires, développées pour plus de 70 langues et listées par la « *Global Wordnet Association* »[2]. Chaque unité de sens de la langue est représentée par un groupe de mots porteur de ce sens (*synset*). Des liens sémantiques tels que l'antonymie (concept opposé), l'hypéronymie (concept plus général) ou l'hyponymie (concept plus spécifique) sont définis entre les *synsets*. Pour le français, on trouve le *Wordnet* Libre du Français (WOLF) (SAGOT et FIŠER 2008) et le projet francophone *EuroWordnet* (VOSSEN 1998). Dans nos travaux et sauf mention contraire, nous utilisons le WOLF pour les traitements sur le français. Les tests ont été réalisés successivement avec les versions 0.1.4 et 0.1.5 publiées en 2009 et 2010. Le problème majeur est la polysémie des mots car si la ressource fournit l'ensemble des sens du mot et les liens existants pour chacun d'eux, il est difficile de déterminer lequel est employé. Dans le cadre de textes longs, la répétition du sens employé dans ceux-ci peut aider à l'identifier, fait non vérifié dans des textes courts.

Soient I^*_{mot} et P^*_{mot} l'importance et la probabilité d'apparition d'un mot selon les diverses stratégies présentées ici, S l'ensemble des *synsets* s de l'ontologie et P_m, la probabilité d'apparition du mot m (10.1).

Nous optons pour des méthodes qui calculent la probabilité d'apparition d'un mot comme :

2. référence : `http://www.globalwordnet.org` (le 28/08/2012)

- la somme des probabilités d'apparition de la racine du mot (formule 10.2) ;
- la somme des probabilités d'apparition de la racine des lemmes du mot (formule 10.3) ;
- la somme des probabilités d'apparition des lemmes partageant un de leurs sens en commun avec les lemmes possibles du mot (formule 10.4) ;
- la somme des probabilités d'apparition de la racine des lemmes partageant un de leurs sens en commun avec les lemmes ayant la même racine que le mot (formule 10.5) ;.

$$\begin{aligned} P_{mot}^{rac} &= \sum_{m \in M} P_m \mid racinisation(m) = racinisation(mot) \\ I_{mot}^{rac} &= -P_{mot}^{rac} * log(P_{mot}^{rac}) \end{aligned} \tag{10.2}$$

$$\begin{aligned} Set(m, mot) &= racinisation(lemmatisation(m)) \cap racinisation(lemmatisation(mot)) \\ P_{mot}^{lem/rac} &= \sum_{m \in M} P_m \mid Set(m, mot) \neq \emptyset \\ I_{mot}^{lem/rac} &= -P_{mot}^{lem/rac} * log(P_{mot}^{lem/rac}) \end{aligned} \tag{10.3}$$

$$\begin{aligned} Syn_{mot}^{lem} &= \{lem \in s \mid \exists l' \in s, l' \in lemmatisation(mot)\} \\ P_{mot}^{lem/senses} &= \sum_{lem \in Syn_{mot}^{lem}} P_{lem} \\ I_{mot}^{lem/senses} &= -P_{mot}^{lem/senses} * log(P_{mot}^{lem/senses}) \end{aligned} \tag{10.4}$$

$$\begin{aligned} Set(l', mot) &= racinisation(l') \cap racinisation(lemmatisation(mot)) \\ Syn^{rac}_{mot} &= \{lem \in s \mid \exists l' \in s, Set(l', mot) \neq \emptyset\} \\ P^{rac/senses}_{mot} &= \sum_{lem \in Syn^{rac}_{mot}} P_{lem} \\ I^{rac/senses}_{mot} &= -P^{rac/senses}_{mot} * log(P^{rac/senses}_{mot}) \end{aligned} \tag{10.5}$$

Les formules 10.4 et 10.5 considèrent que les mots sont porteurs de tous leurs sens possibles quel que soit le contexte d'emploi. Cela permet de ne pas avoir recours à une désambiguïsation au préalable des sens des mots et maximise la probabilité P^*_{mot} utilisée par l'entropie pour chaque mot. Cette stratégie, peu justifiée dans un contexte pluri disciplinaire, l'est dans notre cas car la probabilité d'emploi polysémique d'un terme au sein des textes du corpus se trouve considérablement réduite en l'absence de domaines multiples.

Le tableau 10.1 montre les mots identifiés comme les plus importants dans deux corpus de textes différents selon cette méthode, mots propices aux regroupements d'opinions (ouverture de magasins, progression du chiffre d'affaire, formation à l'anglais, favoriser la mobilité, développer les connaissances culturelles, ...). Pour chaque colonne de mots, la colonne correspondante $I^{rac/senses}_{mot}$ présente l'importance ainsi calculée de ces mots. Les mots « améliorer » et « promouvoir » ont reçu une note identique grâce à la détection d'une proximité sémantique dans leurs sens. De même, les mots « formation »/« formations », « culturel »/« culturelle » ont la même importance grâce à la lemmatisation, et « culture »/« culturelles », « expatriation »/« expatriés » grâce à la racinisation. Nous expliciterons ensuite les effets sur les regroupements de ces pondérations

Similarité sémantique des mots

Wordnet permet également de tenir compte de la distance sémantique entre les mots lors des regroupements. En l'absence de thèmes multiples, nous considérons

que deux mots sont sémantiquement liés si *Wordnet* connaît un sens à ces mots partageant un lien. Dans le cas où plusieurs liens sont trouvés, le lien le plus fort est retenu. A savoir, par ordre décroissant d'importance, la synonymie forte (les deux mots appartiennent au même *synset*), la synonymie faible (les mots appartiennent à des *synsets* « proches de sens »), l'hypéronymie et la fratrie par hypéronymie (les mots appartiennent à des *synsets* qui ont un *synset* hypéronyme en commun).

Nous déduisons ainsi des relations de similarités entre mots à partir des informations disponibles dans le WOLF. Soit $sim_{mot}(m_1, m_2)$, la similarité entre deux les mots m_1 et m_2 définie par :

$$sim_{mot}(m_1, m_2) : M^2 \rightarrow [0,1]$$

et selon les règles suivantes :

– $sim_{mot}(m_1, m_2) = 1 \Leftrightarrow \exists s \in S, \exists (l_1, l_2) \in s^2, l_1 \in lemmatisation(m_1) \wedge l_2 \in lemmatisation(m_2)$,

Résultat sur le jeu « réussite de l'enseigne TrucMuche »					
mot	$I_{mot}^{rac/senses}$	mot	$I_{mot}^{rac/senses}$	mot	$I_{mot}^{rac/senses}$
magasin/s	0.1535	TrucMuche	0.1357	volume/s	0.1271
ouverture/s	0.1519	augmentation/s	0.1336	expansion	0.1248
france/çais	0.1415	nombre/s	0.1315	com	0.1248
client/s/e/es	0.1396	chiffre d affaire	0.1315	vendus/ues	0.1224
enseigne	0.1377	progresse	0.1315		
international/e	0.1357	formation/s	0.1315		
préférée	0.1357	développement/s	0.1294		
Résultat sur le jeu des « choix stratégiques »					
mot	$I_{mot}^{rac/senses}$	mot	$I_{mot}^{rac/senses}$	mot	$I_{mot}^{rac/senses}$
formation/s	0.1668	international/ale/ales/aux	0.1489	connaissance/s	0.1254
anglais/se	0.1591	échange/s	0.1441	différents/tes/ces	0.1254
mobilité/s	0.1591	langue/s	0.1372	personnel/s	0.1232
favoriser	0.1563	sites	0.1372	améliorer	0.1186
développer/ement	0.1519	étranger/ers/ères	0.1354	promouvoir	0.1186
		culture/es/el/elle/els/elles	0.1316	inter	0.1186
		communication/s	0.1296		

TABLE 10.1 – Extraits de l'importance des mots avec l'entropie de Shannon appliquée sur les sens. Chaque couple de colonnes présente en ligne le lemme et ses formes dérivées et fléchies avec leur poids $I_{mot}^{rac/senses}$ associé.

i.e. si un des lemmes de l'ensemble $lemmatisation(m_1)$ appartient au même *synset* qu'un des lemmes de l'ensemble $lemmatisation(m_2)$;

- $sim_{mot}(m_1, m_2) = 0.9$ si un des lemmes de l'ensemble $lemmatisation(m_1)$ appartient à un *synset* de sens proche de celui d'un des lemmes de l'ensemble $lemmatisation(m_2)$;
- $sim_{mot}(m_1, m_2) = 0.5$ si un des lemmes de l'ensemble $lemmatisation(m_1)$ appartient à un *synset* en relation hyperonymique de celui d'un des lemmes de l'ensemble $lemmatisation(m_2)$ ou inversement ;
- $sim_{mot}(m_1, m_2) = 0.5$ si un des lemmes de l'ensemble $lemmatisation(m_1)$ appartient à un *synset* hyponymique d'un *synset* hyperonymique de celui d'un des lemmes de l'ensemble $lemmatisation(m_2)$.

Cette règle de calcul vise à maximiser la probabilité d'identification d'un lien sémantique et la non-désambiguïsation du contexte.

Traitement de la négation et de l'antonymie

Le traitement de la négation est un problème à cause notamment des formes multiples qu'elle peut prendre, simples (ne...pas, ne...plus, ...) ou complexes (double négation souvent par antonymie), et de sa portée dans le texte (sur une ou plusieurs phrases ou parties de phrases). Cette liste n'est pas exhaustive mais illustre bien les problèmes rencontrés. Alors que les chercheurs travaillant sur l'identification de la thématique des textes n'effectuent aucun traitement particulier et se contentent de traiter les mots de la négation comme des mots ordinaires voire de les inclure à la liste noire, d'autres, travaillant sur la sémantique préconisent une différenciation entre les phrases selon que l'idée exprimée est niée ou non (POIRIER et al. 2008). Traiter la négation en particulier nécessite de se confronter au problème de sa portée. Comme les textes sont très courts, nous pouvons poser l'hypothèse que la négation, si elle existe, peut être appliquée sur l'ensemble du texte sans en dégrader le sens (POIRIER et al. 2008). Cela n'est bien sûr pas vérifié dans des textes complexes (phrases avec conjonctions, ...).

Le traitement de l'antonymie peut être effectué, sur le même modèle que les autres similarités sémantiques de mots, en ayant recours à une ressource linguis-

tique telle que *Wordnet*. Nous notons $l \not\equiv l'$, la relation d'antonymie existante entre les lemmes l et l' selon *Wordnet*. Nous définissons ainsi le nombre d'antonymies entre deux messages v_1 et v_2 par :

$$\begin{aligned} CrossLems(m_1, m_2) &= lemmatisation(m_1) \times lemmatisation(m_2) \\ nbAntonyms(v_1, v_2) &= \left|\left\{m_1 \in v_1 \mid \exists m_2 \in v_2, \exists (l, l') \in CrossLems(m_1, m_2), l \not\equiv l'\right\}\right| \end{aligned} \tag{10.6}$$

i.e. la taille de l'ensemble des mots de v_1 tels que un des lemmes d'un des mots de v_1 appartient à un *synset* antonymique de celui d'un des lemmes d'un des mots de v_2.

De même, nous définissons le profil de l'application :

$$nbNegations : V \rightarrow \mathbb{N}$$

comme le nombre d'expressions simples de la négation figurant dans le message.

Les expressions de la négation sont ensuite retirées de la représentation du message et les antonymes établis comme ayant une similarité de 1.

Distance sémantique des textes

Soient I_{1i} l'importance du i^{e} mot de la phrase 1 et I_{2i} l'importance du i^{e} mot de la phrase 2, calculées selon les formules I^*_{mot} définies à la section 10.1.2 page 176. Soient m_{1i} et m_{2i} les i^{e} mots des messages v_1 et v_2 à comparer. Nous définissons la distance sémantique entre v_1 et v_2 par les formules $sim^*_{text}(v_1, v_2)$ définie aux équations 10.8, 10.9 et 10.10 à partir des équations 10.7.

$$\begin{aligned}
S_{1i} &= \max_{i'=1}^{|v2|} sim_{mot}(m_{1i}, m_{2i'}) \\
S_{2i} &= \max_{i'=1}^{|v1|} sim_{mot}(m_{2i}, m_{1i'}) \\
I_{c1} &= \sum I_{1i} * S_{1i} \\
I_{c2} &= \sum I_{2i} * S_{2i} \\
I_{m1} &= \sum I_{1i} \\
I_{m2} &= \sum I_{2i}
\end{aligned} \tag{10.7}$$

$$Sim_{text}(v_1, v_2) = \begin{cases} \frac{\frac{I_{c1}}{I_{m1}} + \frac{I_{c2}}{I_{m2}}}{2} & \text{, si } (nbNegations(v_1) + nbNegations(v_2) \\ & \quad + nbAntonyms(v_1, v_2))\%2 \\ -\frac{\frac{I_{c1}}{I_{m1}} + \frac{I_{c2}}{I_{m2}}}{2} & \text{, sinon} \end{cases} \tag{10.8}$$

$$Sim_{text}^{\neg neg}(v_1, v_2) = \begin{cases} \frac{\frac{I'_{c1}}{I_{m1}} + \frac{I'_{c2}}{I_{m2}}}{2} & \text{, si } nbAntonyms(v_1, v_2)\%2 \\ -\frac{\frac{I'_{c1}}{I_{m1}} + \frac{I'_{c2}}{I_{m2}}}{2} & \text{, sinon} \end{cases} \tag{10.9}$$

Traitements de représentations de textes				
Négation	**Lemmatisation**	**Racinisation**	**Pondération**	**Distance sémantique des mots**
Ne rien faire	Ne rien faire	Ne rien faire	Equi-pondération simple	Similitude de formes
Black List	*TreeTagger*	SnowBall	*Black List* + Equi-pondération	Similitude de sens
Tags des textes			*Black List* + TFIDF	
			Black List + Shannon avec sens	
			Black List + Shannon	

TABLE 10.2 – Différentes stratégies de traitements de représentations de textes. Chaque colonne donne en ligne les différentes stratégies envisageables pour un traitement donné.

$$
\begin{aligned}
I'_{c1} &= \sum I_{1i} \mid \exists m_2 \in v2, m_{1i} = m_2 \\
I'_{c2} &= \sum I_{2i} \mid \exists m_1 \in v1, m_{2i} = m_1 \\
Sim_{text}^{\neg wordsenses}(v_1, v_2) &= \begin{cases} \frac{\frac{I'_{c1}}{I_{m_1}} + \frac{I'_{c2}}{I_{m_2}}}{2} & \text{, si } (nbNegations(v_1) + nbNegations(v_2))\%2 \\ -\frac{\frac{I'_{c1}}{I_{m_1}} + \frac{I'_{c2}}{I_{m_2}}}{2} & \text{, sinon} \end{cases}
\end{aligned}
\tag{10.10}
$$

10.1.3 Résultats expérimentaux sur les regroupements

Le tableau 10.2 reprend les 120 combinaisons de traitements de représentations de textes évoquées. La combinaison soulignée dans le tableau s'est démarquée dans nos réflexions et tests par son adéquation avec nos attentes. Une analyse des regroupements obtenus au moyen d'une méthode hiérarchique dans cette configuration est présentée dans les tableaux 10.3 et 10.4. Les tests conduits sur une méthode par partitions (K-Moyennes), ont montré des problèmes sémantiques similaires à ceux évoqués ci-après pour la méthode hiérarchique.

Premier extrait : jeu « entreprise »	
Titre de la classe	**Messages**
Proximité, convivialité	Développer (la proximité/la convivialité) (dans l'entreprise/avec et entre les collaborateurs)
Conventions	Convention semestrielle ou annuelle / Convention : réunion de tous les collaborateurs
Groupes de projets	(Travailler en/Créer des groupes de) projets transversaux / Favoriser les groupes de projets
Mettre en place	Mettre en place (des groupes de travail transversaux/des objectifs) / Solliciter des volontaires pour mettre en place des groupes de travail transversaux suivis et reconnus par le manager / Mise en place (d'outils de mesure précise de réalisation des objectifs/de réunions régulières d'information)
Fêter les succès	Savoir fêter les succès / Créer des événements réguliers et varies fêter les succès
Deuxième extrait : jeu « relationnel »	
Titre de la classe	**Messages**
Ecouter, observer	Je regarde et j'écoute mes clients / (Ecouter/Etre à l'écoute/Faire parler/Rêver pour) les clients, les fournisseurs / Rester connecté avec le client et la réalité du terrain / Ecouter son instinct / Je vais voir autre chose dans d'autres métiers
Moins d'opérationnel	(Pouvoir se dégager/Je me dégage) de l'opérationnel / Susciter un contre pouvoir non opérationnel
Veille	Se tenir informé du marché (visite clients, veille) / Je visite mes clients / S'informer, reste en veille
Voyager	Je voyage / Voyager, s'ouvrir au monde / Voyages en interne, échanges avec ses équipes

TABLE 10.3 – Extraits de regroupements par une méthode hiérarchique (1re partie). Chaque ligne donne en 1re colonne le titre de classe et en 2e colonne les verbatims la constituant. Notons que les titres des classes n'ont pas été générés automatiquement mais par un codificateur.

Dans le premier extrait, les idées de « réunion régulière », « proximité, convivialité », « fêter les succès » ont été détectées. L'expression « mise en place » a été considérée comme une opinion plutôt que les compléments de cette expression. Dans le deuxième extrait, les idées « d'écouter et de voir » aussi bien les clients que d'autres métiers et de « voyage » ont été trouvées. Cet exemple montre l'intérêt d'employer des techniques telles que la lemmatisation ou la racinisation : le rapprochement sémantique a ainsi pu être fait entre le nom « voyage » et le verbe « voyager ». Dans le troisième extrait, « projets internationaux » et « projets mondiaux » ont été rapprochés par la détection de liens sémantiques entre les termes « internationaux » et « mondiaux ». Par ailleurs, cet extrait porte sur l'un des deux jeux utilisés pour tester la

Troisième extrait : jeu « choix stratégiques »	
Titre de la classe	**Messages**
Projets internationaux	Faire des projets internationaux / Développer une culture mondiale à l'occasion de projet
Différences culturelles	(Sensibilisation/Sensibiliser le personnel) aux différences culturelles
Connaissance culturelle	Développer la connaissance des (cultures de nos clients/différentes "cultures business"/cultures locales/des cultures des autres pays/différents sites) / Se former à la culture
Ouverture international	(Développer les/Promouvoir l'exercice des) langues étrangères / Développer la maîtrise de la langue anglaise / Apprendre l'anglais / Favoriser l'internationalisation des recrutements
Mobilité	Développer (la mobilité/la mobilité internationale (dans les 2 sens))
Quatrième extrait : jeu « commande internet »	
Titre de la classe	**Messages**
Satisfait produits	comme d'habitude je n'ai pas eu de problème avec les produits / J'ai été ravie par ma commande et surtout la rapidité et le sérieux de l'envoi. / je suis contente de vos colis / Rien à dire tout est parfait!! / Frais de livraison un peu élevés je trouve. Les produits sont conformes et de bonne qualité pour le prix. / Je suis très contente / mercie a vous touts
Insatisfait commande	commande soit disant livrée en 48 heures mais livraison seulement après relance le 24/12 ça fait un peu long heureusement qu'il n'y avair pas d'échange à faire / J'ai passé commande le 12/12 et je n'ai pu récupérer ma commande au relais que le 21/12 au lieu de 48h après car il était livré mais pas enregistré. / Annonce par mail commande livrée au point relai et quand je suis allée la chercher elle n'était pas là. / à ce jour je n'ai toujours pas ma commande. / aucun suivi de cette commande. / je n ai rien a dire je suis tres heureuse de ma commande
Non classé	je crois savoir ma la respnsable du pont relais queles livraisons ne sont effectuées que 2 fois par semaine ce qui repousse le délai de 48h prévu. Que popuvez-vous faire pour remédier à ce soucis??

TABLE 10.4 – Extraits de regroupements par une méthode hiérarchique (2e partie). Chaque ligne donne en 1re colonne le titre de classe et en 2e colonne les verbatims la constituant. Notons que les titres des classes n'ont pas été générés automatiquement mais par un codificateur.

pondération. On retrouve bien les idées faisant l'objet de la plus forte pondération : formation à l'anglais, favoriser la mobilité et le développement des connaissances culturelles. Dans le dernier extrait, notre traitement de la négation en particulier a permis de rapprocher le message « je n'ai pas eu de problèmes avec les produits » du groupe « satisfaction produit ». En revanche, sur des textes plus complexes, notre technique engendre comme prévu, des erreurs de compréhension : le message « je

n ai rien a dire je suis tres heureuse de ma commande » a été rapproché du groupe « insatisfait commande » à cause de la négation qu'il comprend (« n »). « mercie a vous touts » a été bien classé, malgré l'orthographe, en combinant la lemmatisation et la racinisation, mais « je crois savoir ma la respnsable du pont relais... » ne l'a pas été.

Une analyse statistique a été réalisée sur les regroupements produits par cette méthode hiérarchique sur un jeu de tests créé à partir de plusieurs *corpora*. Cette qualité a été évaluée, en mesurant l'écart entre le nombre de groupes attendu et produit, et l'homogénéité de ces derniers. Chaque message est codifié manuellement selon l'idée principale qu'il exprime. L'homogénéité d'une classe correspond au pourcentage de messages codifiés avec l'idée majoritaire de la classe. Si aucune idée n'est majoritaire, la classe est déclarée comme totalement hétérogène. L'homogénéité globale est ensuite calculée comme la moyenne des homogénéités de chaque classe pondérée selon le nombre de messages qu'elle contient. Les deux critères étant complémentaires, on considère un résultat meilleur qu'un autre s'il l'est sur l'un des critères et s'il est au moins aussi bon sur l'autre. On admettra par ailleurs heuristiquement qu'un résultat n'est plus vraiment acceptable en dessous d'une homogénéité moyenne de 30% et au delà d'un écart total de 20 rangs ainsi que le fait qu'un gain de 3 rangs sur le critère d'écart permet de compenser une perte moyenne de 10% sur le critère d'homogénéité. Nous avons pris comme référence la combinaison mise en évidence précédemment (« solution avec tous les traitements »), puis nous l'avons comparée avec quelques autres proches en terme de traitements (Fig. 10.2).

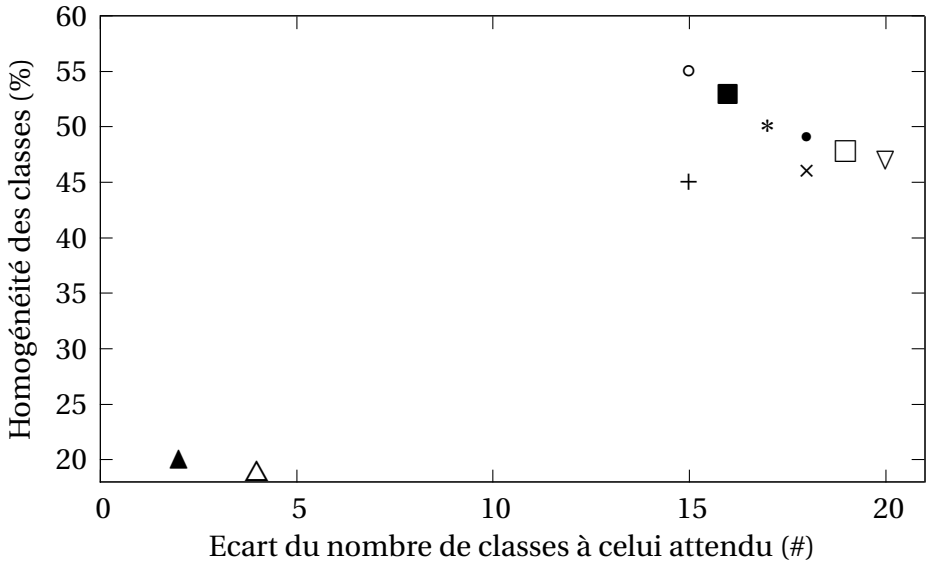
La solution de référence obtient un bon résultat mais les autres traitements de la négation (en liste noire et sans traitement) semblent meilleurs, le traitement en liste noire l'étant davantage que la solution sans traitement. Parmi les messages comportant une négation dans ce jeu, seul 44% répondent aux critères de complexité évoqués. Cela illustre la faible pertinence de cette solution en présence de messages complexes. Il est également intéressant de souligner que toutes les solutions ont produit un nombre de classes plus élevé que celui attendu. Nos solutions n'arriveraient donc pas à regrouper les idées autant qu'un expert. Ce test met également en évidence la très forte pertinence de la liste noire et de la distance sémantique des mots calculée selon les sens. Tous les tests réalisés contradictoirement obtiennent des résultats nettement inférieurs. La pondération des sens semble meilleure que celle des

mots car elle permet de produire moins de groupes même s'ils semblent légèrement moins homogènes. L'emploi de la lemmatisation ou de la racinisation est indispensable sous peine de produire un nombre de classes trop élevé. La lemmatisation se révèle toutefois plus efficace que la racinisation et de qualité presque égale voire légèrement supérieure à la combinaison des deux méthodes. Ce constat tend à prouver que la racinisation peut dégrader les résultats par rapprochements de mots ayant seulement une racine commune.

10.1.4 Conclusion de l'étude préliminaire

Nous avons étudié le regroupement de textes courts d'opinions. Dans ce cadre, la représentation vectorielle se révèle pertinente car elle fournit une distance sémantique simple à calculer et la possibilité d'y adjoindre un système de pondération favorisant les dimensions les plus intéressantes pour les rapprochements. L'entropie de Shannon donne de bons résultats pour le calcul de ces poids surtout après la prise en compte de la sémantique des mots à l'aide d'une ressource lexicale et sans avoir recours à une désambiguïsation du corpus. La recherche de marques de négation en vue d'un marquage des messages dans leur intégralité comme niés ou affirmés nous permet d'obtenir généralement de meilleurs résultats sur les regroupements que les autres solutions envisagées dans le cadre de textes très courts. A l'inverse, cette méthode se révèle médiocre lorsqu'elle est employée sur des messages comportant plusieurs idées. Enfin, la lemmatisation se révèle très peu efficace dans un contexte orthographique difficile. Ce constat n'est pénalisant dans notre application que si les fautes portent sur l'opinion évoquée. La racinisation des formes lemmatisées permet de les corriger dans le cas où la racine n'est pas affectée.

Nous étudions dans les sections suivantes les apports d'une gestion spécifique des messages comportant plusieurs idées ainsi que ceux de la mise en place d'une correction orthographique automatique du corpus avant le traitement.



Symbole	Type	I^*_{mot}	sim^*_{text}	Écart	Homogénéité
▲	Pondération des mots sans liste noire	$I^{lem/rac}_{mot}$	$Sim_{text}(v_1, v_2)$	2	20,12
△	Sans liste noire	$I^{rac/senses}_{mot}$		4	19,94
+	Tous les traitements			15	45,62
∘	Négation en liste noire		$Sim^{\neg neg}_{text}(v_1, v_2)$	15	55,15
■	Sans traitement de la négation			16	53,94
*	Sans racinisation	$I^{lem/senses}_{mot}$	$Sim_{text}(v_1, v_2)$	17	50,91
×	Sans lemmatisation	I^{rac}_{mot}		18	46,83
•	Pondération des mots	$I^{lem/rac}_{mot}$		18	49,1
□	Ni lemmatisation, ni racinisation	I_{mot}		19	48,92
▽	Similitude de forme des mots	$I^{rac/senses}_{mot}$	$Sim^{\neg wordsenses}_{text}(v_1, v_2)$	20	47,83

FIGURE 10.2 – Résultats statistiques de 10 chaînes de traitements. La figure se compose d'un graphique et d'un tableau. Le graphique présente pour chacune des 10 chaînes de traitements testées représentée par un symbole, l'homogénéité en % des classes produites en fonction de l'écart du nombre de classes produites à celui attendu. Le tableau reprend ces même informations en colonne pour chaque chaîne de traitements testée en ligne. Il indique en plus en 3e et 4e colonne la pondération des mots et la distance textuelle utilisées pour chacune.

10.2 Reconnaissance des messages comportant plusieurs idées : étude de leur impact sur l'apprentissage non supervisé

Nous avons fait précédemment l'hypothèse que tous les textes traités ne comportaient qu'une seule idée. Pourtant, même si un texte est court, il n'est pas exclu qu'il comporte plusieurs idées. Ainsi, le verbatim de l'exemple 10.2 (« Le choix, les prix, la qualité »), alors qu'il ne contient que 6 mots, comporte trois idées regroupées dans un thème commun par la taxonomie souhaitable.

Exemple 10.2

Le choix, les prix, la qualité

- *satisfait par les produits*
  - *prix abordable*
  - *large choix*
  - *bonne qualité des produits*

L'hypothèse faite nous a permis d'employer les algorithmes standard d'apprentissage non supervisé tels que les K-moyennes ou l'apprentissage hiérarchique.

Les algorithmes fonctionnent comme nous l'avons vu à la section 8.2 sur le principe de la recherche d'une partition de qualité suffisante sur l'ensemble des données d'entrée. Cela suppose alors un biais fort d'une disjonction entre les classes produites par ces algorithmes. En terme de regroupement de verbatims d'opinions et dans notre contexte applicatif de synthèses d'idées, cela impose de considérer que chaque verbatim n'appartient qu'à une seule et unique classe et ne comporte qu'une seule idée.

10.2.1 Objectif

Il convient donc de pouvoir tolérer l'absence de disjonction entre les classes produites par apprentissage pour les éléments de l'espace d'entrée correspondant à des

verbatims comportant plusieurs idées.

Afin d'atteindre cet objectif, nous devons proposer des techniques et critères permettant de reconnaître ces verbatims problématiques sur lesquels la conjonction de classes sera tolérée et ainsi les différencier des autres qui devront rester disjoints.

10.2.2 Critères de séparation

Notre but est d'identifier des critères capables de nous aider à différencier les verbatims comportant plusieurs idées de ceux n'en comportant qu'une seule.

Un premier axe de réflexion concerne le risque lié à une mauvaise reconnaissance de la catégorie d'un verbatim. S'il est reconnu comme comportant une seule idée alors qu'il en possède plusieurs, l'algorithme d'apprentissage ne pourra distinguer les différentes idées qu'il contient. Au contraire, s'il est reconnu comme possédant plusieurs idées alors qu'il n'en possède qu'une, l'algorithme d'apprentissage sera autorisé à tort à effectuer une conjonction de parties sur ce verbatim. Il y aura alors un risque que le verbatim soit classé dans plusieurs classes de façon erronée.

Cette réflexion induit un déséquilibre entre les deux risques encourus : si dans le premier cas, l'algorithme est forcé de considérer comme n'exprimant qu'une seule idée un verbatim en exprimant plusieurs, dans le deuxième cas, il est simplement autorisé à considérer la présence de plusieurs idées sans que cela ne soit imposé. Il est ainsi préférable de reconnaître à tort un verbatim avec une seule idée comme en comportant plusieurs que l'inverse. Cela ne nous empêche bien sûr pas d'être aussi précis que possible dans la reconnaissance des verbatims comportant plusieurs idées.

Dans nos travaux, trois critères différents sont envisagés afin de caractériser les verbatims comportant plusieurs idées par rapport à ceux n'en comportant qu'une seule :

1. La taille du verbatim. Plus un verbatim est long et plus il apparaît comme raisonnable de considérer qu'il est composé de plusieurs idées.
2. Le nombre de verbes. Plus un verbatim contient un nombre élevé de verbes et plus il est plausible que plusieurs idées différentes sont développées.

3. La présence de mots de liaison. La présence de conjonctions de coordination dans le verbatim augmente la vraisemblance qu'il énonce plusieurs idées liées les unes aux autres par les conjonctions.

En raison de la syntaxe peu rigoureuse que nous avons à traiter, le troisième critère n'a pas été étudié dans nos travaux. Notre étude porte donc sur les deux autres et leur pertinence dans la reconnaissance des verbatims comportant plusieurs idées.

10.2.3 Optimisation des critères

Nous cherchons à optimiser les critères mentionnés afin de distinguer au mieux les verbatims comportant plusieurs idées.

Pour ce faire, nous utilisons un jeu de messages constitué de :

– 118 verbatims au total

– 77 verbatims comportant une seule idée (65.25%)

– 41 verbatims comportant plusieurs idées (34.75%)

Étudions d'abord le critère de taille des verbatims. Soit $L \in \mathbb{N}*$ la valeur du paramètre indiquant la limite choisie pour la taille du message entraînant sa reconnaissance en tant que verbatim comportant plusieurs idées. En d'autre termes, si $L = l$, un verbatim contenant l mots ou plus est considéré comme comportant plusieurs idées contrairement à un verbatim en possédant moins. La figure 10.3 présente différents résultats obtenus sur la reconnaissance des verbatims comportant plusieurs idées en fonction de ce critère.

Intéressons-nous d'abord aux erreurs de reconnaissance des verbatims comportant plusieurs idées. Il est évident qu'en prenant une valeur faible pour le paramètre, nous diminuons le risque de ne pas reconnaître un tel verbatim en comparaison d'une valeur de paramètre élevée. Ainsi, pour $L = 1$, aucune erreur n'est commise dans cette reconnaissance tandis que pour $L = 11$, près de 61% des verbatims comportant plusieurs idées ne sont pas détectés.

Au contraire, la reconnaissance des verbatims ne comportant qu'une seule idée est compromise pour une valeur du paramètre trop faible en comparaison d'une valeur plus élevée. Ainsi, pour $L = 1$, l'intégralité des verbatims comportant une seule

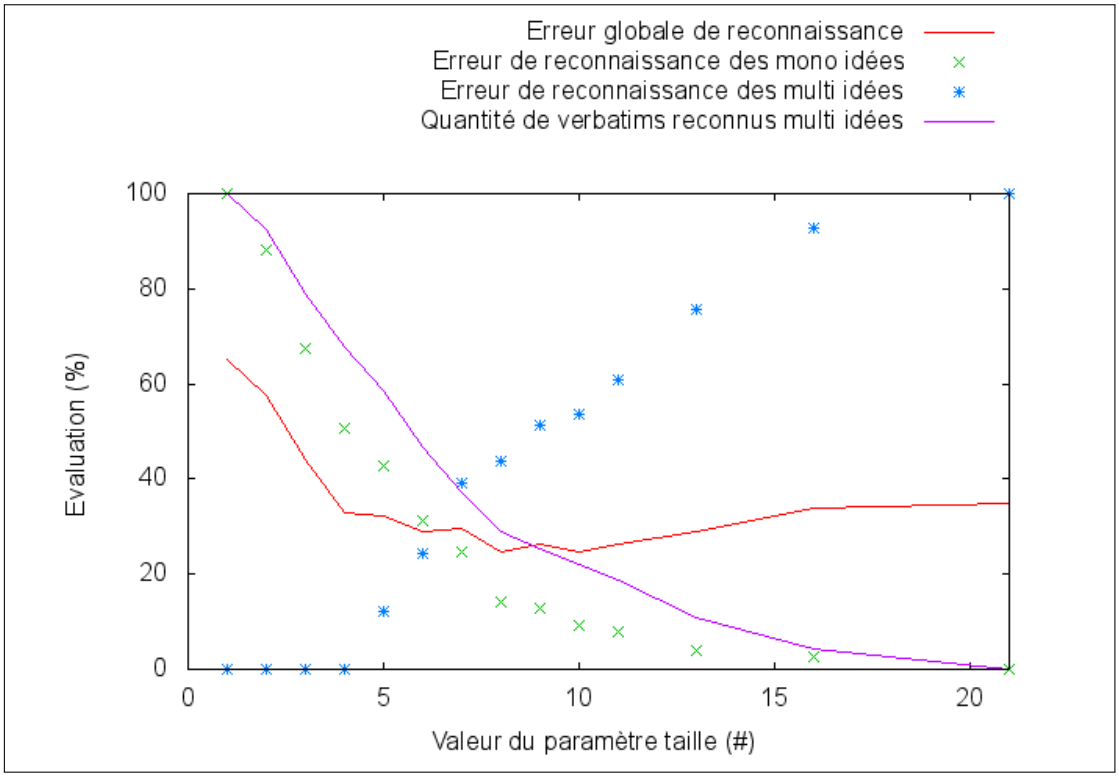


FIGURE 10.3 – Erreurs de reconnaissance des verbatims comportant une ou plusieurs idées en fonction de la valeur du paramètre taille

idée ne sont pas reconnus tandis que pour $L = 11$, seuls 8% des verbatims comportant une seule idée ne sont pas détectés.

Globalement, plus la valeur du paramètre augmente et moins les verbatims sont reconnus comme comportant plusieurs idées.

Le meilleur compromis est trouvé pour une valeur moyenne du paramètre : $L = 6$. A cette valeur, 47% des verbatims sont reconnus comme comportant plusieurs idées. 31% des verbatims ne comportant qu'une seule idée ne sont pas détectés tandis que 24% des verbatims comportant plusieurs idées ne le sont pas.

Notons par ailleurs que le taux d'erreur moyen peut être diminué davantage en prenant une valeur de paramètre à $L = 8$. Cependant, pour cette valeur, 44% des verbatims comportant plusieurs idées ne sont pas reconnus contre seulement 24% pour $L = 6$. Comme nous l'avons souligné auparavant, cette perte de reconnaissance des verbatims comportant plusieurs idées n'est pas souhaitable même si cela nous prive de la reconnaissance de quelques verbatims ne comportant qu'une seule idée : seul 14% des verbatims ne comportant qu'une seule idée ne sont pas reconnus pour $L = 8$ contre 31% pour $L = 6$.

Étudions ensuite le critère du nombre de verbes dans les verbatims. Soit $V \in \mathbb{N}*$ la valeur du paramètre indiquant la limite choisie pour le nombre de verbes dans le message entraînant sa reconnaissance en tant que verbatim comportant plusieurs idées. La figure 10.4 présente différents résultats obtenus sur la reconnaissance des verbatims comportant plusieurs idées en fonction de ce critère.

Là encore, nous observons une décroissance du nombre de verbatims reconnus comme comportant plusieurs idées allant de 78% pour $V = 1$ à moins d'1% pour $V = 11$. On observe également le même compromis entre les erreurs réalisées sur la reconnaissance des verbatims ne comportant qu'une seule idée et sur ceux qui en comportent plusieurs. Le meilleur compromis est atteint pour $V = 3$. A cette valeur, 19% des verbatims ne comportant qu'une seule idée ne sont pas détectés et 39% des verbatims en comportant plusieurs ne le sont pas non plus pour un taux d'erreur global de 26%.

Le tableau 10.5 donne une comparaison entre les évaluations obtenues avec le critère taille et le critère nombre de verbes.

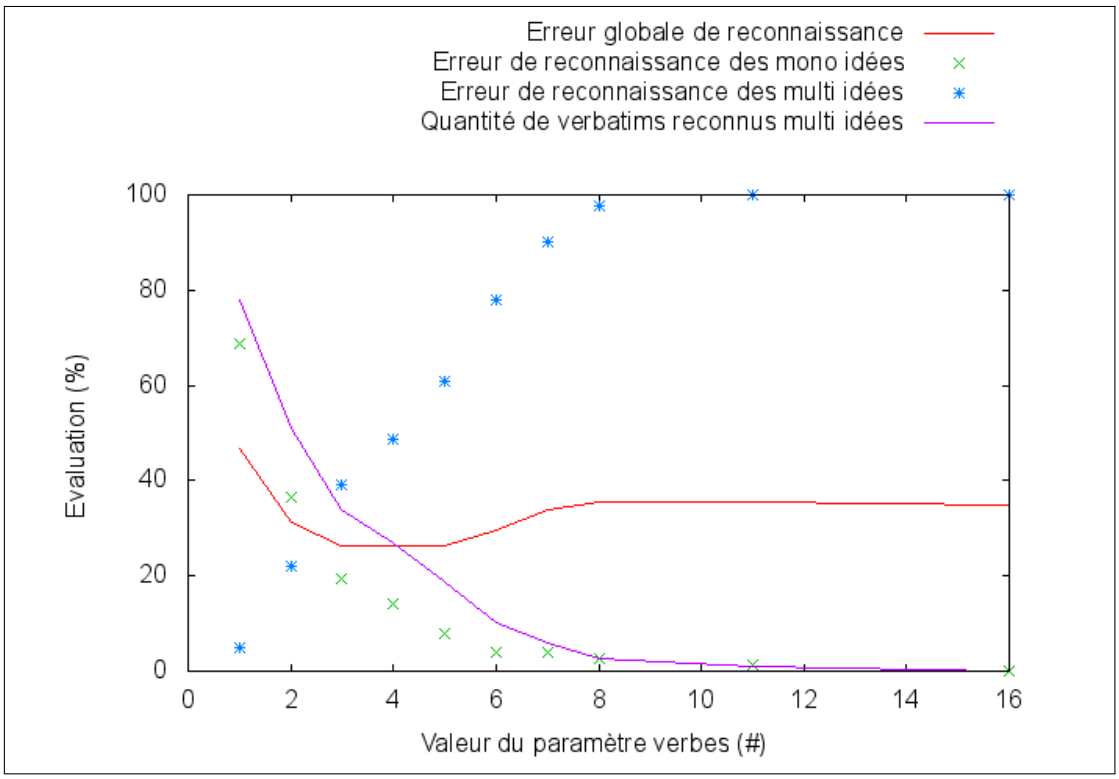


FIGURE 10.4 – Erreurs de reconnaissance des verbatims comportant une ou plusieurs idées en fonction de la valeur du paramètre nombre de verbes

Critère(s)	**erreur sur les verbatims contenant une seule idée**	**erreur sur les verbatims comportant plusieurs idées**
taille ($L = 6$)	31	24
nombre de verbes ($V = 3$)	19	39
Gain/Perte	-12	+15

TABLE 10.5 – Erreurs sur la reconnaissance des verbatims comportant une ou plusieurs idées avec les critères taille ($L = 6$) et nombre de verbes ($V = 3$). En colonne sont présentées les erreurs sur les verbatims contenant respectivement une et plusieurs idées pour les critères en ligne respectivement de la taille et du nombre de verbes.

Notons qu'aux deux valeurs de compromis trouvées ($L = 6$ et $V = 3$), le paramètre s'appuyant sur la taille des verbatims commet moins d'erreurs sur la reconnaissance des verbatims comportant plusieurs idées que le paramètre s'appuyant sur le nombre de verbes : 24% des verbatims comportant plusieurs idées ne sont pas détectés pour $L = 6$ tandis que 39% des verbatims en comportant plusieurs ne le sont pas pour $V = 3$. Dans le même temps, le critère reposant sur le nombre de verbes commet moins d'erreurs dans la reconnaissance des verbatims ne comportant qu'une seule idée que le critère reposant sur la taille : 19% d'erreurs pour $V = 3$ contre 31% d'erreurs pour $L = 6$.

Aucun des deux critères n'est donc meilleur que l'autre si l'on considère une égalité sur les deux types d'erreurs commises. Cependant, nous avons vu qu'il nous fallait sanctionner plus sévèrement les erreurs sur les verbatims comportant plusieurs idées. Cela nous conduit à privilégier le critère de la taille qui conduit à 15% d'erreurs en moins.

Étudions ensuite l'apport de chacun des deux critères à l'autre.

Commençons par l'apport du critère du nombre de verbes au critère de la taille des verbatims. Fixons la valeur du paramètre nombre de verbes à $V = 3$. La figure 10.5 présente différents résultats obtenus sur la reconnaissance des verbatims comportant plusieurs idées en fonction du critère taille du verbatim pour $V = 3$.

Le meilleur compromis est atteint pour une valeur du paramètre $L = 7$. A cette valeur, 29% des verbatims comportant une seule idée ne sont pas reconnus ainsi que 29% des verbatims en comportant plusieurs. Le tableau 10.6 donne une comparaison entre les évaluations obtenues avec le critère taille seul et le critère taille ajouté au critère du nombre de verbes.

Par rapport au meilleur compromis obtenu avec le critère de la taille des verbatims seul où 31% des verbatims ne comportant qu'une seule idée n'étaient pas détectés et 24% des verbatims comportant plusieurs idées ne l'étaient pas non plus, nous obtenons une amélioration de 2% sur les verbatims comportant une seule idée pour une dégradation de 5% sur les verbatims en comportant plusieurs.

La dégradation importante sur la reconnaissance des verbatims comportant plusieurs idées nous pousse à conclure que l'apport du critère du nombre de verbes à la

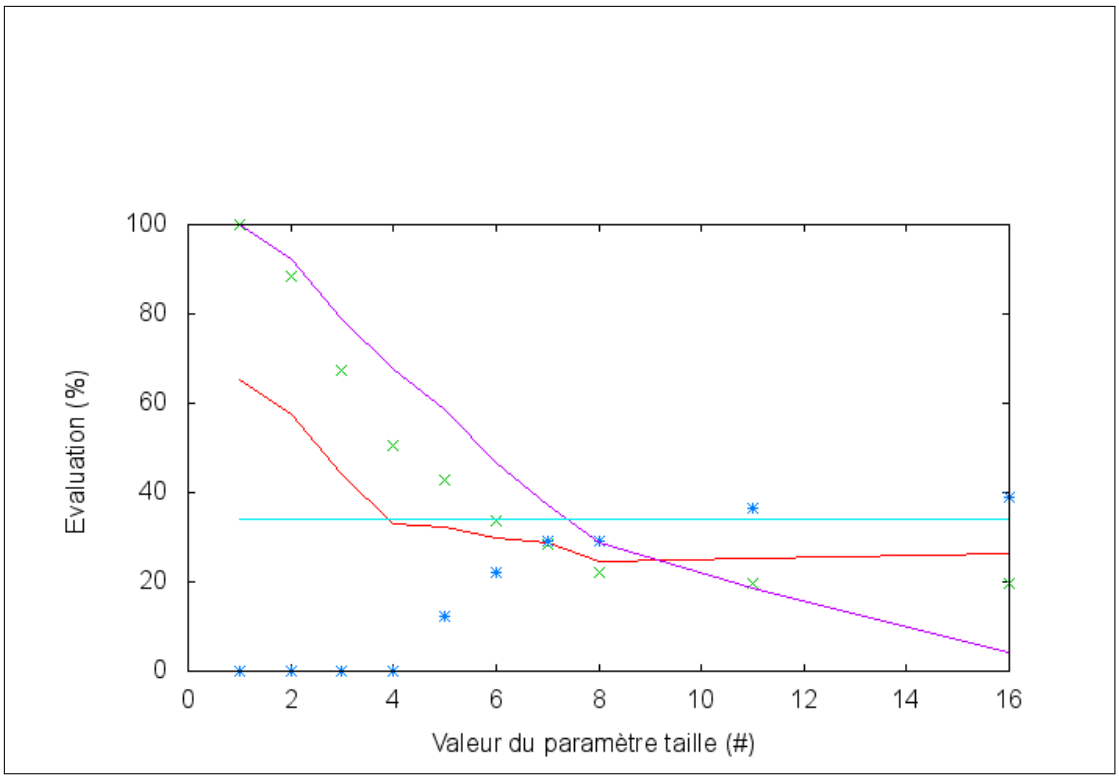


FIGURE 10.5 – Erreurs de reconnaissance des verbatims comportant une ou plusieurs idées en fonction de la valeur du paramètre taille pour $V = 3$

Critère(s)	**erreur sur les verbatims contenant une seule idée**	**erreur sur les verbatims comportant plusieurs idées**
taille seule ($L = 6$)	31	24
taille + nombre de verbes ($L = 7$ et $V = 3$)	29	29
Gain/Perte	-2	+5

TABLE 10.6 – Erreurs sur la reconnaissance des verbatims comportant une ou plusieurs idées entre le critère taille seul ($L = 6$) et l'association des critères taille et nombre de verbes ($L = 7$ et $V = 3$). En colonne sont présentées les erreurs sur les verbatims contenant respectivement une et plusieurs idées pour les critères en ligne respectivement de la taille seule et de l'association de la taille et du nombre de verbes.

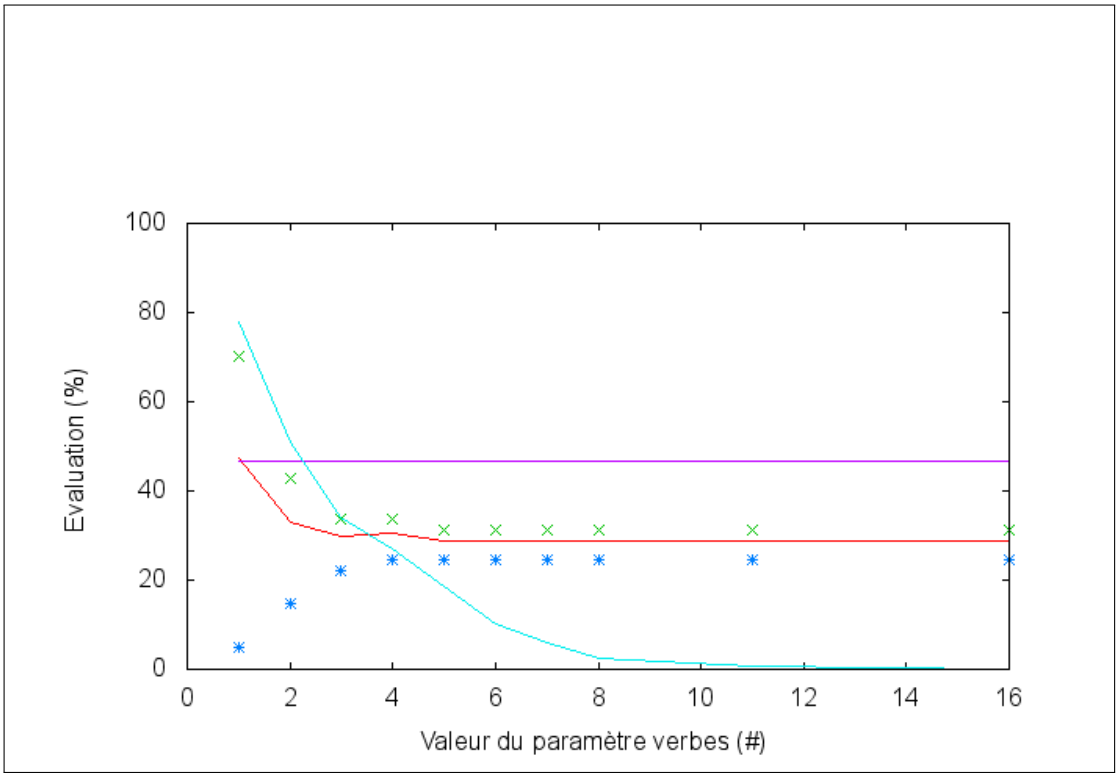


FIGURE 10.6 – Erreurs de reconnaissance des verbatims comportant une ou plusieurs idées en fonction de la valeur du paramètre nombre de verbes pour $L = 6$

taille n'est pas concluant.

Étudions ensuite l'apport du critère de la taille des verbatims au nombre de verbes. Fixons la valeur du paramètre taille des verbatims telle que $L = 6$. La figure 10.6 présente différents résultats obtenus sur la reconnaissance des verbatims comportant plusieurs idées en fonction du critère nombre de verbes pour $L = 6$.

Le meilleur compromis est atteint pour une valeur du paramètre $V = 5$. A cette valeur, 31% des verbatims comportant une seule idée ne sont pas reconnus ainsi que 24% des verbatims en comportant plusieurs. Le tableau 10.7 donne une comparaison entre les évaluations obtenues avec le critère nombre de verbes seul et le critère nombre de verbes ajouté au critère de la taille des verbatims.

Par rapport au meilleur compromis obtenu avec le critère du nombre de verbes seul où 19% des verbatims ne comportant qu'une seule idée n'étaient pas détectés et 39% des verbatims comportant plusieurs idées ne l'étaient pas non plus, nous avons

Critère(s)	**erreur sur les verbatims contenant une seule idée**	**erreur sur les verbatims comportant plusieurs idées**
nombre de verbes seul ($V = 3$)	19	39
nombre de verbes + taille ($V = 5$ et $L = 6$)	31	24
Gain/Perte	+12	-15

TABLE 10.7 – Erreurs sur la reconnaissance des verbatims comportant une ou plusieurs idées entre le critère nombre de verbes seul ($V = 3$) et l'association des critères taille et nombre de verbes ($V = 5$ et $L = 6$). En colonne sont présentées les erreurs sur les verbatims contenant respectivement une et plusieurs idées pour les critères en ligne respectivement du nombre de verbes seul et de l'association de la taille et du nombre de verbes.

Critère(s)	**erreur sur les verbatims contenant une seule idée**	**erreur sur les verbatims comportant plusieurs idées**
taille seule ($L = 6$)	31	24
nombre de verbes + taille ($V = 5$ et $L = 6$)	31	24
Gain/Perte	0	0

TABLE 10.8 – Erreurs sur la reconnaissance des verbatims comportant une ou plusieurs idées entre le critère taille seul ($L = 6$) et l'association des critères taille et nombre de verbes ($V = 5$ et $L = 6$). En colonne sont présentées les erreurs sur les verbatims contenant respectivement une et plusieurs idées pour les critères en ligne respectivement de la taille seule et de l'association de la taille et du nombre de verbes.

donc une amélioration de 15% sur les verbatims comportant plusieurs idées pour une dégradation de 12% sur les verbatims n'en comportant qu'une seule.

Comme le présente le tableau 10.8, l'ajout du critère taille au critère nombre de verbes rapproche les meilleurs taux d'erreurs rencontrés de ceux obtenus pour le critère de taille seul. La simplicité du critère taille seul nous conduit alors à préférer cette solution.

10.2.4 Conclusion sur l'optimisation

Nos évaluations nous conduisent à considérer le critère de la taille des verbatims comme le meilleur des deux critères retenus. De plus, nous avons vu que l'apport du critère du nombre de verbes à celui-ci est négatif, nous conduisant donc à écarter l'association des deux critères.

La meilleure valeur trouvée pour ce paramètre est identifiée à $L = 6$.

10.2.5 Évaluation du critère pour la reconnaissance de verbatims comportant plusieurs idées

Nous devons à présent nous assurer que considérer la valeur du paramètre taille des verbatims comme optimal pour $L = 6$ est satisfaisant pour tout ensemble de verbatims à traiter. Il est en fait évident que le résultat optimal est différent si l'ensemble de verbatims à traiter l'est puisque cette propriété est inhérente aux verbatims de l'ensemble. Le but ici est donc de vérifier que l'erreur commise en considérant une monotonie de cette propriété d'un ensemble à traiter à l'autre est suffisamment faible pour être négligeable. Pour cela, nous proposons d'effectuer la même évaluation que celle faite sur l'ensemble d'optimisation pour le paramètre de la taille des verbatims sur un nouvel ensemble de verbatims que nous dénotons ensemble de validation.

Le jeu de validation comporte :

– 38.15% de verbatims comportant une seule idée

– 61.85% de verbatims comportant plusieurs idées

Notons le changement important dans le ratio du nombre de verbatims comportant plusieurs idées et ceux n'en possédant qu'une entre l'ensemble utilisé pour l'optimisation et l'ensemble d'évaluation présenté ici. Nous avons environ une diminution de moitié du pourcentage de verbatims comportant une seule idée pour un pourcentage doublé de verbatims en comportant plusieurs.

La figure 10.7 présente les résultats obtenus.

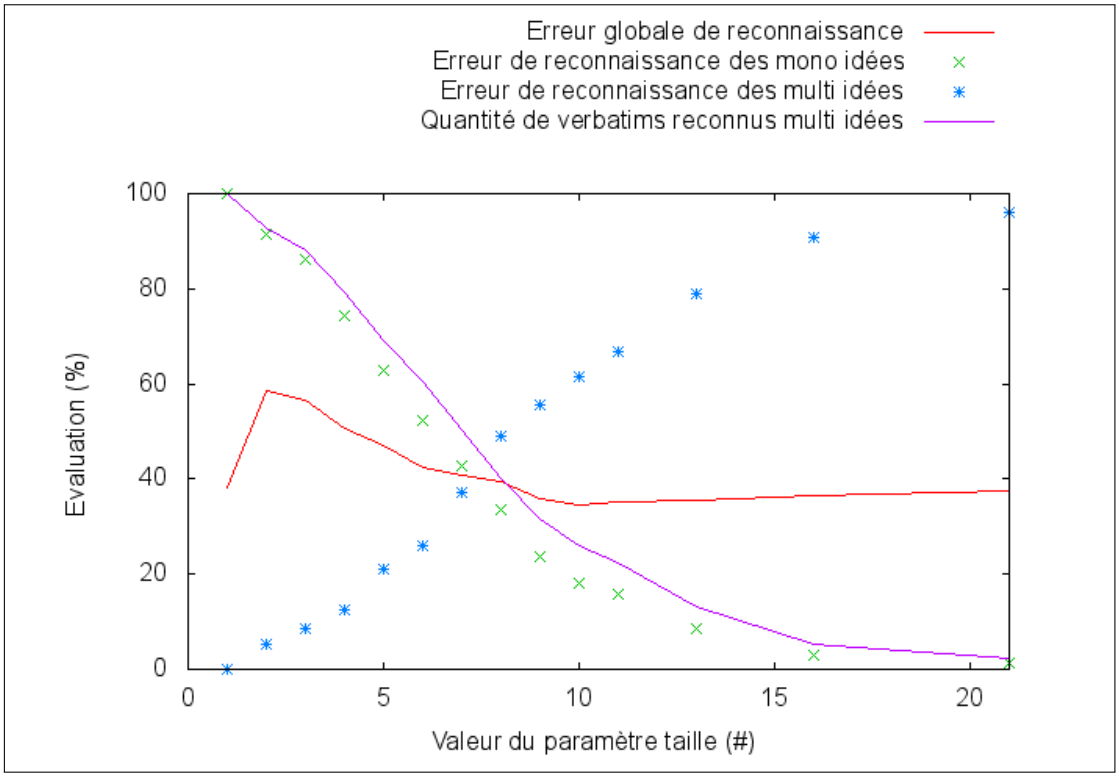


FIGURE 10.7 – Erreurs de reconnaissance des verbatims comportant une ou plusieurs idées en fonction de la valeur du paramètre taille pour l'ensemble de validation

Le meilleur compromis est atteint pour la valeur $L = 7$. A cette valeur, 43% des verbatims comportant une seule idée et 37% des verbatims en comportant plusieurs ne sont pas détectés pour un taux d'erreur global de 41%. La valeur optimale du paramètre n'a donc que faiblement évoluée en comparaison du changement important dans les ratios.

Considérer la valeur optimale du paramètre comme fixe a donc un impact limité sur la qualité du système.

10.2.6 Évaluation du critère pour l'apprentissage de verbatims comportant plusieurs idées

Nous ne devons bien sûr pas oublier le but final de la reconnaissance des verbatims comportant plusieurs idées : permettre une meilleure classification de ceux-ci en permettant aux algorithmes d'effectuer un recouvrement sur ces données. Nous devons pour cela définir une stratégie de traitement spécifique à ces verbatims.

Nous optons pour un traitement en deux temps des verbatims selon leur reconnaissance comme comportant une ou plusieurs idées :

1. Les verbatims reconnus comme comportant une seule idée sont traités par l'algorithme d'apprentissage non supervisé hiérarchique classique.
2. La classification précédemment obtenue est utilisée pour constituer un ensemble d'apprentissage. Cet ensemble sert à entrainer un algorithme de proche voisinage modifié pour classer les verbatims comportant plusieurs idées. Cet algorithme de proche voisinage est utilisé pour classer les verbatims reconnus comme comportant plusieurs idées.

Algorithme des K-plus proches voisins : adaptation à la classification de verbatims comportant plusieurs idées

Nous employons une version modifiée de l'algorithme des K-plus proches voisins afin de permettre la reconnaissance de plusieurs idées. Un espace de voisinage est défini autour du verbatim à classer selon un rayon fixé. Si ce voisinage contient des verbatims de l'échantillon d'apprentissage, l'ensemble formé par l'union de toutes

les classes affectées à ces verbatims d'apprentissage est affecté au verbatim à classer en tant que classes de ce dernier. Si cet espace ne contient aucun des verbatims de l'échantillon d'apprentissage, le verbatim n'est pas classé (i.e., il ne possède aucune idée reconnue). La stratégie de fixer le rayon de l'espace de voisinage est motivé par les hautes dimensions de notre espace de travail. Ces conditions peuvent, en effet, imposer à l'algorithme de rechercher assez loin dans l'espace le voisin le plus proche d'un verbatim à classer. Dans ce cas, il n'est plus garanti que la validité bayésienne soit respectée du fait que l'espace de voisinage considéré augmente.

Ainsi, cette règle de classification des verbatims revient à attribuer à un point x de l'espace d'entrée toutes les classes i telles que $k_i \neq 0$ dans le voisinage retenu. Nous avons de plus vu à la section 7.2.1 que :

$$k_i = mVP(i|x)p(x).$$

D'où

$$k_i \neq 0 \Leftrightarrow P(i|x) \neq 0$$

puisque m, V et $p(x)$ sont différents de 0.

Cette règle de classification peut être considérée comme relativement optimiste (i.e., elle classe de manière optimiste un verbatim dans une classe i) puisqu'un verbatim est affecté à une classe si la probabilité locale estimée qu'il appartienne à cette classe n'est pas nulle. Cela nous oblige à considérer un espace de voisinage suffisamment restreint pour limiter le nombre de classes respectant cette condition aux plus probables. Autrement, la précision de notre système en serait altérée.

Nous supposons ici le rayon de voisinage, noté R_N, idéal pour une valeur de 0.6. Nous réaliserons dans la suite de l'étude une optimisation plus précise de ce paramètre.

Résultat de l'évaluation

Nous avons évalué notre système d'apprentissage avec reconnaissance des verbatims comportant plusieurs idées et comparé les résultats obtenus avec ceux pro-

duits par l'algorithme d'apprentissage hiérarchique classique appliqué à l'ensemble des verbatims d'évaluation.

L'ensemble d'évaluation est constitué de :

– 561 verbatims au total

– 179 verbatims comportant une seule idée (31.9%)

– 382 verbatims comportant plusieurs idées (68.1%)

En terme de ratio de verbatims comportant plusieurs idées, nous sommes assez proches sur cet ensemble du premier ensemble de validation.

Nous proposons d'utiliser les mêmes deux critères d'évaluation que ceux utilisés lors de l'étude préliminaire (section 10.1.3), à savoir l'homogénéité moyenne des classes et l'écart avec le nombre de classes attendues. La figure 10.8 présente les résultats obtenus sur ces deux critères pour l'ensemble d'évaluation en fonction de la valeur du paramètre taille L. Elle présente également le résultat obtenu par ce même ensemble d'évaluation sur les deux critères en employant un algorithme hiérarchique classique.

Notons d'abord que la quasi intégralité des évaluations réalisées en prenant en compte la reconnaissance des verbatims comportant plusieurs idées obtiennent de meilleurs résultats que l'évaluation de l'algorithme hiérarchique classique appliqué à l'ensemble des verbatims de l'évaluation. Seules les valeurs les plus faibles du paramètre ($L = 1$ et $L = 2$) constituent des exceptions.

Le meilleur résultat est atteint pour $L = 4$ avec une homogénéité de 48.7% et un écart du nombre de groupes par rapport à celui attendu de 11.

10.3 Représentation des mots pour une correction orthographique hybride basée sur les caractères et les phonèmes

Le deuxième axe de travail mis en évidence par l'étude préliminaire concerne la nécessité de procéder à une correction orthographique des textes à traiter. Ces derniers sont, en effet, saisis la plupart du temps dans des contextes n'incitant pas à une

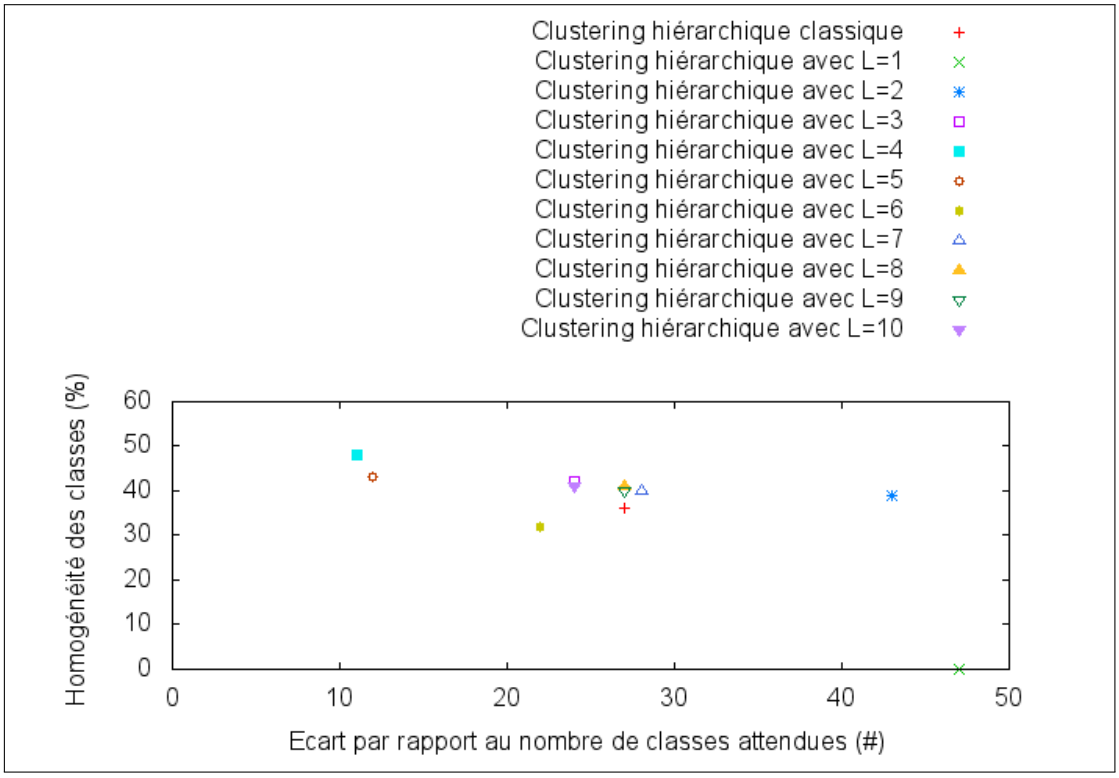


FIGURE 10.8 – Homogénéité des classes obtenues en fonction de l'écart par rapport au nombre de classes attendues. Le graphique présente pour chacune des 11 configurations testées représentée par un symbole, l'homogénéité en % des classes produites en fonction de l'écart du nombre de classes produites à celui attendu. 10 de ces configurations comportent la reconnaissance des verbatims comportant plusieurs idées. La 11^{e} correspond à un traitement sans aucune reconnaissance.

grande rigueur sur ce point. Le pire des cas étant certainement lorsque les textes sont saisis dans des questionnaires disponibles sur Internet.

Nous sommes alors confrontés à plusieurs types d'erreurs d'orthographe :

- Erreurs de saisies
  - une lettre d'une touche voisine est saisie à la place ou en plus de la lettre attendue
  - une lettre attendue est oubliée

    Exemple 10.3

    Pas d'attete, des prix clairs une réduction importante

  - deux lettres sont inversées
- Écriture phonétique
  - les mots sont écrits de manière phonétique. Cela conduit parfois à obtenir un mot existant similaire phonétiquement au mot correct

    Exemple 10.4

    produits bien mais chair (surtout la déco ! !)

    Exemple 10.5

    le tant d'attante

Il apparaît important de gérer ces différentes erreurs possibles. Pour cela, notre système de correction s'appuie sur des représentations basées sur les caractères et les phonèmes des mots afin de déterminer le meilleur mot possible dans chaque situation.

Nous commençons notre étude par une description de la représentation choisie des mots par leurs phonèmes.

10.3.1 Représentation des mots par leurs phonèmes

Nous avons présenté au chapitre 4 plusieurs algorithmes permettant la représentation phonétique des mots. Nous avons ainsi décrit les algorithmes :

- Soundex

- Soundex 2
- Metaphone
- Double Metaphone
- Phonex

Même si un encodage numérique de la suite de phonèmes ne nous semble pas un choix judicieux dans nos travaux, l'algorithme Phonex s'est particulièrement démarqué par une prise en compte qui se veut très rigoureuse des spécificités de la langue française dans sa partie visant à établir la suite de phonèmes à partir de la suite de caractères. Il constitue de ce fait une base intéressante pour notre travail.

Nous proposons d'utiliser une amélioration de cet algorithme visant à construire une suite de phonèmes à partir de la suite de caractères. Les différences entre notre algorithme et celui de l'algorithme Phonex, tel que décrit à la section 4.2.5, apparaissent en fonte grasse.

1. **Passage en casse minuscule et remplacement des caractères typographiques spéciaux à, â, ä, î, ï, ù, û, ü, ô, ö, ë par leurs homologues sans diacritique**
2. Remplacement des y par des i
3. Remplacement du ph par f
4. **Remplacement du x par z si précédé et suivi de a, e, i, o, u, é, è, ê**
5. **Remplacement du x par ske si précédé de a, e, i, o, u, é, è, ê**
6. **Remplacement des s par des z s'ils sont suivi et précédés des lettres a, e, i, o, u, é, è, ê**
7. **Remplacement du c par un s s'il est suivi des lettres e, é, è, ê ou i**
8. **Remplacement du ç par un s**
9. Remplacement des groupes de lettres suivantes :

Groupe de lettres	gan	gam	gain	gaim
Correspondant Phonex	kan	kam	kain	kaim

10. Remplacement des occurrences suivantes, si elles sont suivies par une lettre a, e, i, o, u, **é, è, ê ou par la dernière lettre du trigramme**

Groupe de lettres	ain	ein	aim	eim
Correspondant Phonex	yn	yn	**ym**	**ym**

11. Remplacement des groupes de 3 lettres suivants (sons 'o', 'ein') :

Groupe de lettres	eau	ein	ain	eim	aim
Correspondant Phonex	o	4	4	4	4

12. Remplacer les sons 'é' **finaux**

Groupe de lettres	ée	ées	ère	ères	er	et
Correspondant Phonex	y	y	y	y	y	y

13. **Remplacement du son 'é' en général**

Groupe de lettres	é	è	ê	ai	ei
Correspondant Phonex	y	y	y	y	y

14. **Remplacement du son 'é' si suivi de deux consonnes identiques**

Groupe de lettres	e
Correspondant Phonex	y

15. Remplacement des groupes de 2 lettres suivantes (sons 'an', 'in' **et 'un'**), sauf s'ils sont suivis par une lettre a, e, i o, u, y, un son 1 à 4 **ou la même consonne que la finale**

Groupe de lettres	an	am	en	em	in	im	**un**	**um**
Correspondant Phonex	1	1	1	1	4	4	**4**	**4**

16. **Prise en compte du son "on" sauf s'il est suivi par une lettre a, e, i o, u, y, un son 1 à 4 ou la même consonne que la finale**

Groupe de lettres	on	om
Correspondant Phonex	6	6

17. Remplacement des groupes de 2 lettres suivants

Groupe de lettres	oe	eu	au	oi	oy	**oua**	ou
Correspondant Phonex	e	e	o	2	2	**2**	3

18. Remplacement des groupes de lettres suivants

Groupe de lettres	ch	sch	sh	ss
Correspondant Phonex	5	5	5	s

19. Remplacement des lettres ou groupe de lettres suivants dans l'ordre du tableau

Groupe de lettres	c	qu	q	gu	ga	go	g
Correspondant Phonex	k	k	k	k	ka	ko	j

20. **Suppression des h restants**

21. Suppression **récursive** des terminaisons suivantes : t, x, **d, p et s sauf les terminaisons "et" et "er" (son 'é')**

Groupe de lettres	**et**	**er**
Correspondant Phonex	**y**	**y**

22. Suppression des lettres dupliquées (doublons)

10.3.2 Représentation des mots par leur squelette consonantique

Nous utilisons les méthodes Soundex et Soundex2 pour représenter les squelettes consonantiques des mots. Nous obtenons ainsi deux squelettes consonantiques pour chaque mot. Notons $soundex(mot)$ et $soundex2(mot)$, les deux squelettes consonantiques de mot.

10.3.3 Représentation des mots par leurs caractères

Les mots sont représentés en tant que suites de caractères comme présenté à la section 4.2.2. Notons de plus que la casse est ignorée mais pas les diacritiques.

Exemple 10.6

Le tableau suivant donne quelques exemples de nos choix de représentations de mots par suites de caractères :

Mot	***Représentation du mot par suite de caractères selon nos choix***
maison	*(m, a, i, s, o, n)*
Maison	*(m, a, i, s, o, n)*
tache	*(t, a, c, h, e)*
tâche	*(t, â, c, h, e)*

Chaque ligne donne le mot à représenter en 1re colonne et sa représentation en 2e colonne.

10.3.4 Distance hybride basée sur les deux types de représentations

Nous souhaitons établir une distance hybride s'appuyant sur les deux types de représentations vus au préalable. Cette distance hybride doit notamment permettre

Phonème 1	Phonème 2	Coût de substitution du phonème 1 par le 2 et inversement
e	y	0.5
-	-	1

TABLE 10.9 – Coût de substitution des phonèmes pour la distance de Levenshtein. La 1re colonne donne le phonème d'origine et la 2e colonne le phonème de substitution. La 3e et dernière colonne donne le coût de substitution du phonème d'origine par celui de substitution et inversement. Le caractère joker '-' représente n'importe quel phonème.

d'évaluer la distance orthographique la plus juste entre deux chaînes de caractères l'une valide d'un point de vue orthographique et l'autre non. Nous devons en outre proposer une technique permettant de déterminer une distance maximale en dessous de laquelle la chaîne de caractères non valide d'un point de vue orthographique pourra être considérée comme équivalente à celle valide.

La distance de Levenshtein, présentée à la section 4.3.4 est un choix de base judicieux pour établir une distance entre les deux représentations par suites de caractères et les deux représentations phonétiques. Nous proposons dans cette section des améliorations à la distance classique et une méthode permettant de concilier les deux distances obtenues afin de prendre une décision quant à la correction orthographique d'une chaîne de caractères erronée par une chaîne valide.

Adaptation de la distance de Levenshtein pour la représentation par phonèmes

Une première adaptation se situe au niveau des coûts de substitutions. Le tableau 10.9 donne les coûts de substitution que nous utilisons pour la distance de Levenshtein entre les phonèmes.

L'adaptation nécessaire ici ne concerne que les deux phonèmes 'e' et 'y' qui voient leur coût de substitution diminué de moitié par rapport aux autres substitutions.

Une seconde adaptation est nécessaire au niveau des coûts d'insertion et de suppression. La valeur de la case $d[i, j]$ de la matrice de Levenshtein est donnée par la

Phonème 1	**Phonème 2**	**Coût d'insertion et de suppression du phonème 1 si précédé par le 2**
consonne	consonne	0.7
-	-	1

TABLE 10.10 – Coût d'insertion et de suppression des phonèmes dans la distance de Levenshtein. La 1re colonne donne le phonème inséré ou supprimé et la 2e colonne le phonème précédant le 1er. La 3e et dernière colonne donne le coût d'insertion et de suppression du 1er phonème si précédé par le 2e. Le caractère joker '-' représente n'importe quel phonème.

formule :

$$\min d[i-1,j] + \text{ coût de suppression }, d[i,j-1] + \text{ coût d'insertion }, d[i-1,j-1] + \text{ coût de substitution } - |\text{ coût de suppression } - \text{ coût d'insertion }|$$

Notons $dist_{pho}(chaîne1, chaîne2)$, la distance phonétique obtenue entre les deux chaînes de caractères après phonétisation.

Adaptation de la distance de Levenshtein pour la représentation par caractères

Pour l'ajout et la suppression de caractères, le coût utilisé est le coût classique de 1 pour une distance de Levenshtein. Notre adaptation se situe au niveau des coûts de substitutions. Le tableau 10.11 donne les coûts de substitution que nous utilisons pour la distance de Levenshtein entre les caractères.

L'adaptation nécessaire ici est plus importante que pour les phonèmes. La fonction VoisinDe(?) présente dans le tableau correspond à la fonction renvoyant tous les caractères « voisins » d'un caractère donné en paramètre. La notion de voisinage utilisée ici correspond à une proximité immédiate de deux lettres sur un clavier. Pour le français, nous considérons les voisinages sur un clavier de type AZERTY. L'exemple 10.7 donne des exemples de voisinages de caractères.

Caractère 1	**Caractère 2**	**Coût de substitution du caractère 1 par le 2 et inversement**
é	è	0.5
é	ê	0.5
é	e	0.5
è	ê	0.5
è	e	0.5
ê	é	0.5
ê	e	0.5
à	â	0.5
à	a	0.5
â	a	0.5
î	i	0.5
c	ç	0.5
?	VoisinDe(?)	0.7
-	-	1

TABLE 10.11 – Coût de substitution des caractères dans la distance de Levenshtein. La 1re colonne donne le caractère d'origine et la 2e colonne le caractère de substitution. La 3e et dernière colonne donne le coût de substitution du caractère d'origine par celui de substitution et inversement. Le caractère joker '-' représente n'importe quel caractère.

Exemple 10.7

Caractère	***Caractères voisins sur un clavier AZERTY***
'a'	*{'q', 'z', 's'}*
'g'	*{'r', 't', 'y', 'h', 'b', 'v', 'f'}*

Notons $dist_{car}(chaîne1, chaîne2)$, la distance d'édition ainsi calculée entre deux chaînes de caractères.

Distance entre représentations consonantiques

Nous avons mis en évidence que les algorithmes Soundex et Soundex 2 réduisent beaucoup la taille de l'espace de représentation des mots par rapport à l'espace d'origine. Il nous paraît donc inopportun de considérer une relation de distance permettant d'effectuer des rapprochements complémentaires par programmation dynamique comme le permettrait une distance de Levenshtein.

Nous choisissons donc d'employer simplement une distance 0/1 telle que :

$$dist_{con}(chaîne1, chaîne2) = \begin{cases} 0 & \text{ssi} \quad \begin{cases} soundex(chaîne1) = soundex(chaîne2) \\ \text{et} \\ soundex2(chaîne1) = soundex2(chaîne2) \end{cases} \\ 1 & \text{sinon} \end{cases}$$

Correction orthographique hybride

Nous devons à présent concilier trois distances afin de prendre une décision sur la correction orthographique d'une chaîne erronée notée $chaîneerror$. Soient $chaîne1$ et $chaîne2$, deux chaînes valides d'un point de vue de l'orthographe en compétition pour être substituées à la chaîne erronée. L'algorithme 1 renvoie la chaîne ou les chaînes choisies pour être substituées à la chaîne erronée.

Si la chaîne erronée ne contient que des consonnes, alors la distance consonantique est utilisée. Si la distance consonantique est nulle avec l'une des chaînes candidates, celle-ci est retournée. Si la distance est nulle avec les deux chaînes, l'ordre alphabétique est utilisé pour départager. Sinon, l'ensemble vide est retourné. Dans le cas où la chaîne erronée contient des voyelles, les distances d'édition par caractères et par phonèmes sont employées. Si les deux distances sont d'accord pour privilégier l'une ou l'autre des deux chaînes, celle-ci est retournée. Dans le cas contraire, si pour l'une des deux chaînes, la plus petite des deux distances avec la chaîne erronée est inférieure aux deux distances entre l'autre chaîne candidate et la chaîne erronée, alors elle est retournée. Sinon, l'ordre alphabétique est utilisé pour départager.

Input : $erreur$: mot erroné ; $prop_1$, $prop_2$: propositions à départager
Output : la meilleure proposition

```
1  begin
2      if containsOnlyConsonants(erreur) then
3          if dist_con(erreur, prop_1) = 0 ∧ dist_con(erreur, prop_2) = 1 then
4              return prop_1;
5          end
6          if dist_con(erreur, prop_2) = 0 ∧ dist_con(erreur, prop_1) = 1 then
7              return prop_2;
8          end
9          if dist_con(erreur, prop_1) = 0 ∧ dist_con(erreur, prop_2) = 0 then
10             return lower_alpha_order(prop_1, prop_2);
11         end
12         return ∅;
13     end
14     if dist_car(erreur, prop_1) <
       dist_car(erreur, prop_2) ∧ dist_pho(erreur, prop_1) < dist_pho(erreur, prop_2)
       then
15         return prop_1;
16     end
17     if dist_car(erreur, prop_2) <
       dist_car(erreur, prop_1) ∧ dist_pho(erreur, prop_2) < dist_pho(erreur, prop_1)
       then
18         return prop_2;
19     end
20     d ← min{dist_car(erreur, prop_1), dist_pho(erreur, prop_1)};
21     if d < dist_car(erreur, prop_2) ∧ d < dist_pho(erreur, prop_2) then
22         return prop_1;
23     end
24     d ← min{dist_car(erreur, prop_2), dist_pho(erreur, prop_2)};
25     if d < dist_car(erreur, prop_1) ∧ d < dist_pho(erreur, prop_1) then
26         return prop_2;
27     end
28     return lower_alpha_order(prop_1, prop_2);
29 end
```

Algorithme 1 : Algorithme de correction orthographique. Si la chaîne erronée ne contient que des consonnes, la distance consonantique seule est utilisée. Sinon, la proposition meilleure que l'autre à la fois pour la distance phonétique et pour la distance d'édition des caractères ou à défaut celle obtenant une meilleure distance pour ces 2 distances que la meilleure distance obtenue par l'autre proposition est choisie.

10.3.5 Évaluations de la correction orthographique

Premières évaluations des propositions et décision de correction

Notre approche basée sur une évaluation de la distance entre les mots erronés et les mots corrects pour effectuer la correction présente des avantages et des inconvénients connus.

Le principal avantage de cette démarche réside dans sa robustesse à traiter des textes fortement bruités. Cette force réside dans le fait que la méthode n'emploie pas le contexte des mots erronés pour les corriger. Cela se révèle particulièrement judicieux si les mots contextuels sont eux aussi erronés.

L'inconvénient réside dans le fait que pour un mot erroné, une multitude de mots corrects peuvent être trouvés dans un voisinage très proche d'un point de vue de la phonétique et des caractères les constituant. Cet inconvénient est dû paradoxalement à l'avantage cité au préalable.

Exemple 10.8

> *Prenons l'exemple du mot erroné « chato ». Les mots « château » et « chaton » apparaissent tous deux comme des corrections probables l'un d'un point de vue phonétique et l'autre d'un point de vue de la suite de caractères constituant les mots. Seule une analyse en contexte dans la phrase peut lever une telle ambiguïté.*

Nous devons dès lors adopter une stratégie nous permettant de sélectionner le mot juste parmi l'ensemble des mots les plus probables identifiés par la distance hybride.

Nous devons d'abord déterminer à partir de cette distance hybride, l'ensemble des mots que nous considérons comme les plus probables. Pour cela, nous constituons trois ensembles différents de mots erronés en provenance de trois traitements différents. Pour chacun d'eux, nous associons manuellement les mots erronés à leur correspondant correct attendu selon les sens des messages traités. Nous mesurons alors sur chacun de ces ensembles de test le pourcentage des ensembles proposés pour chaque forme erronée contenant le mot correct attendu. Cette étude est réalisée en considérant des tailles d'ensembles proposés différents. Le tableau 10.12 donne les résultats obtenus sur ces évaluations.

Ensemble de test	**Taille de l'ensemble des propositions**						
	1	**3**	**6**	**9**	**12**	**15**	**30**
Ensemble 1	70%	81%	86%	90%	93%	93%	93%
Ensemble 2	56%	76%	80%	84%	86%	88%	90%
Ensemble 3	65%	83%	87%	88%	88%	91%	91%
Moyenne	64%	80%	84%	87%	89%	91%	91%
Variation de la moyenne	64%	16%	4%	3%	2%	2%	0%
Variation minimale	56%	11%	4%	1%	0%	0%	0%
Variation maximale	70%	20%	5%	4%	3%	3%	2%

TABLE 10.12 – Évaluation des propositions de correction. Le tableau donne pour chacun des trois ensembles de test, le taux des ensembles proposés pour chaque mot erroné contenant la proposition correcte. Les taux sont donnés pour diverses tailles des ensembles proposés par le correcteur. Le tableau donne également la moyenne de ces taux pour l'ensemble des 3 tests pour une même taille de l'ensemble des propositions. Les variations entre deux tailles d'ensembles de propositions successifs sont données pour la moyenne des trois tests ainsi que pour les tests les moins et les plus variants

L'inconvénient annoncé de notre méthode de correction apparaît dans ce test au travers d'un pourcentage moyen de propositions correctes par rapport à celles attendues lorsque les ensembles de propositions se limitent à une seule proposition. Seuls 64% des propositions faites en moyenne sont celles attendues.

Le pourcentage moyen d'ensembles de propositions contenant la proposition attendue croît rapidement pour atteindre 80% environ avec 3 propositions dans chaque ensemble. Il peut encore être augmenté pour passer à 87% pour 9 propositions dans chaque ensemble. Au delà de 9 propositions, le taux d'ensembles contenant la proposition attendue continue d'augmenter très légèrement pour atteindre 91% à 15 propositions.

L'augmentation du nombre de propositions même s'il permet d'augmenter les taux d'ensembles contenant la bonne proposition contribue également à complexifier le problème en demandant au système de choisir au final une des propositions et donc d'analyser chacune d'entre elles. De plus, la possibilité que le choix du système se porte finalement sur une autre proposition que celle attendue augmente également avec le nombre de propositions.

Ainsi, devoir analyser jusqu'à 9 propositions pour un seul mot erroné constitue dans notre cas où une grande partie des mots du corpus contiennent des fautes d'orthographe, une augmentation de la tâche importante. L'augmentation de la moyenne des résultats de nos tests n'est que de 7% par rapport aux ensembles à 3 propositions. De plus, la possibilité que le choix du système se porte finalement sur une autre proposition que celle attendue est triplée par rapport aux ensembles de 3 propositions. Ce constat nous pousse à ne pas opter pour une taille d'ensembles de propositions à 9.

Choisir entre les ensembles à 1 ou 3 propositions est plus délicat. Opter pour 3 propositions permet d'augmenter significativement le taux d'ensembles contenant la bonne proposition (+16%) mais triple dans le même temps les choix que doit faire le système et donc triple le risque d'erreur.

Nous optons donc dans notre étude pour des ensembles restreints à 1 propositions en raison de la simplicité de la décision finale du système qui est alors binaire : remplacer ou non le mot erroné par la proposition.

Nous devons à présent définir plus précisément la fonction de décision binaire permettant au système d'accepter ou non la proposition de correction faite.

Il est assez courant dans le domaine d'avoir recours à une analyse syntaxique de la phrase afin d'éliminer les propositions ne correspondant pas à la syntaxe attendue du langage. Cependant, en raison de la syntaxe peu rigoureuse des textes à traiter, une validation par la statistique globale du corpus nous parait ici plus judicieuse.

Ainsi, la correction d'un mot erroné par la proposition faite est acceptée si et seulement si le mot proposé est présent par ailleurs dans le corpus des textes à traiter. Dans le cas contraire, le mot erroné est conservé car considéré comme plus fiable statistiquement que la proposition de correction.

Évaluation de l'impact de la correction sur les regroupements

Nous évaluons à présent l'impact de la correction orthographique telle que décrite dans les paragraphes précédents sur la qualité des regroupements obtenus.

Nous utilisons pour cela deux ensembles d'évaluation distincts que nous dénotons ensemble 1 et ensemble 2. Le tableau 10.13 donne les résultats obtenus par l'al-

Statistique	**Ensemble de test**			
	Ensemble 1		**Ensemble 2**	
	Correction orthographique			
	sans	**avec**	**sans**	**avec**
Nombre de verbatims	561		197	
Nombre de classes attendus	47		36	
Nombre de mots corrigés	-	322	-	105
Nombre de mots corrigés en moyenne par verbatim	-	0.57	-	0.53
Homogénéité des classes (%)	48.70	48.23	24.15	25.12
Nombre de classes trouvées	36	33	16	17
Écart du nombre de classes trouvées par rapport à celui attendu	11	14	20	19

TABLE 10.13 – Évaluation de l'impact des corrections sur la qualité des regroupements obtenus en apprentissage. Le tableau présente en ligne différents critères permettant d'évaluer les regroupements obtenus sur les deux ensembles présentés en colonne. Pour chacun des deux ensembles, le tableau donne par ailleurs le comparatif des critères avec et sans la correction afin de juger de l'impact de celle-ci.

gorithme de regroupement sur les deux ensembles avec et sans la correction orthographique.

Les deux ensembles de test ont des volumes différents en terme de verbatims. Le premier contient 561 verbatims tandis que le second en contient 197. Nous corrigeons en outre 322 mots dans les 561 verbatims du premier ensemble et 105 mots dans les 197 verbatims du deuxième. Ramené à un ratio de nombre de mots corrigés par rapport au nombre de verbatims, cela donne les deux taux très proches de 0.57 et 0.53 mots corrigés en moyenne par verbatim.

En ce qui concerne l'évolution des regroupements par l'ajout de la correction orthographique, les résultats obtenus sur le premier ensemble de test sont négatifs puisque l'on constate une légère perte sur les critères d'homogénéité et l'écart du nombre de classes obtenues et attendues. L'homogénéité passe de 48.70% sans la correction à 48.23% avec la correction, soit une perte de 0.47%. Dans le même temps, l'écart du nombre de classes trouvés par rapport à celui attendu passe de 11 classes sans la correction à 14 classes avec la correction, soit une perte de 3 classes sur ce critère.

Les résultats obtenus sur le deuxième ensemble sont eux positifs puisque l'on constate cette fois un léger gain sur les deux mêmes critères. L'homogénéité passe de 24.15% sans la correction à 25.12% avec la correction, soit un gain de 0.97%. Dans le même temps, l'écart du nombre de classes trouvés par rapport à celui attendu passe de 20 classes sans la correction à 19 classes avec la correction, soit un gain d'une classe sur ce critère.

Les résultats obtenus en terme de qualité de regroupements par la correction orthographique sont mitigés. La correction peut apporter un gain comme c'est la cas pour le deuxième ensemble de test mais peut également dégrader le résultat comme le montre les résultats obtenus sur le premier ensemble de test.

Cette variation de la qualité ne semble pas corrélée à une variation du taux de correction du corpus puisque les deux ensembles de tests ont subi la même correction moyenne approximative de 0.5 mots corrigés par verbatim. Une explication peut se trouver dans l'importance des mots corrigés pour les regroupements. Un mot corrigé correctement ayant un impact fort sur les regroupements améliore la qualité des regroupements tandis qu'un mot corrigé même correctement mais ayant peu d'impact sur les regroupements n'améliorera pas la qualité mesurée.

Évaluation à grande échelle du système de correction sur le corpus WiCoPaCo

Les premières évaluations du correcteur sont améliorables sur trois points :

1. elles sont réalisées uniquement sur de petits jeux de tests. Une évaluation à large échelle est intéressante à mettre en place.
2. les jeux de tests sont réalisés par nos soins. Un jeu de test « externe » permettrait d'avoir une vision plus globale sur les performances de l'outil.
3. aucune comparaison n'est faite avec une autre stratégie ou outil de correction orthographique.

Le projet WiCoPaCo (MAX et WISNIEWSKI 2010)(WISNIEWSKI, MAX et YVON 2010) vise à constituer un corpus de révisions de Wikipédia. La dernière version proposée est la version 2 et constitue notre base de travail. A partir de celui-ci, un corpus de fautes d'orthographes a pu être extrait (WISNIEWSKI, MAX et YVON 2010). Ce corpus, disponible dans sa version 3, comprend :

Mot erroné	**Mot corrigé**	**Mot erroné**	**Mot corrigé**
lithurgique	liturgique	dicutées	discutées
championn	champion	fourage	fourrage
reellement	réellement	moyeb	moyen
ordianaire	ordinaire	lithanienne	lituanienne
ingèré	ingéré	figurationv	figuration
Genvices	Gencives	Commuauté	Communauté
répendu	répandues	alonger	allonger
répendu	répandu	realiser	réaliser
thétrale	théâtrale	enlevèent	enlevèrent
fracale	fractale	dateant	datent
corynthien	corinthien	estt	est
chorégraaphies	chorégraphies	passéder	posséder
visaches	viscaches	Evasion	Évasion
virsus	versus	culimant	culminant

TABLE 10.14 – Exemples de corrections de « non mots ». Les 1re et 3e colonnes donnent le mot erroné n'appartenant pas au lexique de la langue et les 2e et 4e colonnes le mot juste correspondant appartenant au lexique de la langue.

- 138 873 entrées correspondantes à des corrections en contexte, dont
  - 83 548 entrées correspondantes à des corrections de mots réels en contexte ;
  - 55 325 entrées correspondantes à des corrections de non mots en contexte.

Le tableau 10.14 donne quelques exemples de corrections de « non mots » ainsi extraites du WiCoPaCo.

Des tests ont été conduits sur les 55 325 corrections des « non mots » et sont présentés dans le tableau 10.15.

Puisque notre système de correction ne tient pas compte du contexte des mots, une simplification a été réalisée en vue de supprimer les doublons correspondant à des contextes différents. Il en résulte :

- 63 140 corrections distinctes, dont
  - 33 673 corrections distinctes de mots réels ;
  - 29 467 corrections distinctes de non mots.

Des tests similaires aux premiers, dont les résultats sont présentés à la table 10.16, ont été réalisés sur les 29 467 corrections distinctes des « non mots ».

Nombre de mots corrigés justes à la proposition					**Total**
nº 1	**nº 2**	**nº 3**	**nº 4**	**nº 5**	
28 984	3 024	1 422	606	387	
52.39%	5.47%	2.57%	1.10%	0.70%	35197
nº 6	**nº 7**	**nº 8**	**nº 9**	**nº 10**	63.62%
241	142	183	139	69	
0.44%	0.26%	0.33%	0.25%	0.12%	

TABLE 10.15 – Performances de notre correcteur sur les corrections de « non mots » extraits du WiCoPaCo (version 3 du corpus de correction). Le 1er groupe de colonnes donne le nombre de mots corrigés justes à différentes propositions. La dernière colonne donne le nombre de mots total corrigés justes.

Nombre de mots corrigés justes à la proposition					**Total**
nº 1	**nº 2**	**nº 3**	**nº 4**	**nº 5**	
15 933	1 776	947	439	313	
54.07%	6.03%	3.21%	1.49%	1.06%	20 007
nº 6	**nº 7**	**nº 8**	**nº 9**	**nº 10**	67.90%
200	103	105	125	66	
0.68%	0.35%	0.36%	0.42%	0.22%	

TABLE 10.16 – Performances de notre correcteur sur les corrections distinctes de « non mots » extraits du WiCoPaCo (version 3 du corpus de correction). Le 1er groupe de colonnes donne le nombre de mots corrigés justes à différentes propositions. La dernière colonne donne le nombre de mots total corrigés justes.

Nombre de mots corrigés justes à la proposition					**Total**
nº 1	**nº 2**	**nº 3**	**nº 4**	**nº 5**	
28 063	2 920	1 377	590	375	
52.27%	5.44%	2.56%	1.10%	0.70%	34071
nº 6	**nº 7**	**nº 8**	**nº 9**	**nº 10**	63.46%
233	136	176	135	66	
0.43%	0.25%	0.33%	0.25%	0.12%	

TABLE 10.17 – Performances de notre correcteur sur les corrections de « non mots » extraits du WiCoPaCo (version 2 du corpus de correction). Le 1er groupe de colonnes donne le nombre de mots corrigés justes à différentes propositions. La dernière colonne donne le nombre de mots total corrigés justes.

Dans (WISNIEWSKI, MAX et YVON 2010), des tests qualitatifs de diverses stratégies de corrections sont réalisés à l'aide du corpus de corrections extraites du WiCoPaCo. Ces tests n'ont cependant pas été réalisés sur la dernière version du corpus de corrections mais à partir de la version 2 qui comprend :

- 146 593 entrées correspondantes à des corrections en contexte, dont
  - 92 906 entrées correspondantes à des corrections de mots réels en contexte ;
  - 53 687 entrées correspondantes à des corrections de non mots en contexte.

Des tests similaires aux précédents, dont les résultats sont présentés à la table 10.17, ont été réalisés sur les 53 687 corrections des « non mots ».

La formule de qualité suivante est utilisé pour évaluer les approches de correction orthographique avec le WiCoPaCo :

$$qualité = \frac{1}{N} \sum_{i=1}^{N} \frac{d_i}{s_i}$$

La somme se fait sur les N exemples de l'ensemble d'évaluation, d_i vaut 1 si la i^e correction est dans l'ensemble des suggestions, 0 sinon et s_i est la taille de l'ensemble de suggestions correspondant.

Le tableau 10.18 donne les résultats qualitatifs obtenus par notre correcteur comparé à d'autres sur les « non mots » du WiCoPaCo.

Nous observons que notre correcteur est en terme qualitatif meilleur que toutes les stratégies proposées dans (WISNIEWSKI, MAX et YVON 2010), qui ne tiennent pas

Stratégie de correction	Notre correcteur	Hunspell (H)	Ensembles de confusions (EC)	Patrons (P)	H + EC + P
Qualité	$\frac{30306}{53687}$ = 56.45%	40%	51.1%	35.5%	38.9%

TABLE 10.18 – Qualité de notre correcteur comparée à des systèmes de correction hors contexte sur les « non mots » du WiCoPaCo. La 1re ligne donne différentes stratégies de correction présentées dans (WISNIEWSKI, MAX et YVON 2010) ainsi que la nôtre. La seconde ligne donne pour chaque stratégie la note de qualité relevée dans (WISNIEWSKI, MAX et YVON 2010) ou dans nos propres tests pour notre stratégie de correction.

Stratégie de correction	Notre correcteur	3-grammes	5-grammes
Qualité	$\frac{30306}{53687}$ = 56.45%	58.9%	75.2%

TABLE 10.19 – Qualité de notre correcteur comparée à des systèmes de correction en contexte sur les « non mots » du WiCoPaCo. La 1re ligne donne différentes stratégies de correction présentées dans (WISNIEWSKI, MAX et YVON 2010) ainsi que la nôtre. La seconde ligne donne pour chaque stratégie la note de qualité relevée dans (WISNIEWSKI, MAX et YVON 2010) ou dans nos propres tests pour notre stratégie de correction.

compte du contexte des mots erronés.

En revanche, les travaux de (WISNIEWSKI, MAX et YVON 2010) incluent également des tests de correction incluant le contexte des mots erronés par le biais d'un apprentissage sur des représentations n-grammes. Même s'il n'est pas explicitement précisé dans (WISNIEWSKI, MAX et YVON 2010) s'il s'agit de n-grammes de mots ou de lettres, nous pouvons supposer au vu des travaux cités ((CARLSON et FETTE 2007) et (ISLAM et INKPEN 2009)) utilisant des n-grammes de mots qu'il s'agit de n-grammes de mots. La méthode consiste en un apprentissage visant à extraire d'une partie du WiCoPaCo, une corrélation statistique entre les co-occurrents apparaissant dans les n-grammes des mots erronés et leur correction. Ce modèle statistique est ensuite utilisé pour réordonner les candidats proposés par la méthode $H + EC + P$ (cf. tableau 10.18).

Le tableau 10.19 donne les résultats qualitatifs obtenus par notre correcteur comparé aux approches n-grammes sur les « non mots » du WiCoPaCo.

Nous voyons que notre correcteur est en terme qualitatif :

– sensiblement équivalent à la méthode 3-grammes de (WISNIEWSKI, MAX et YVON 2010) ;
– nettement moins bon que la méthode 5-grammes de (WISNIEWSKI, MAX et YVON 2010).

Cependant, 46.6% des corrections de « non mots » présentes dans le corpus sont des corrections redondantes (i.e., la même correction apparaît au moins deux fois dans le corpus). Cela peut entraîner une forte redondance entre les corrections apprises et testées par les stratégies de correction par apprentissage depuis le corpus. Outre les stratégies par n-grammes, ce biais a une incidence, par exemple, sur la méthode de correction par apprentissage d'ensembles de confusions (CARLSON et FETTE 2007) qui obtient ainsi un score de 51.1% dans (WISNIEWSKI, MAX et YVON 2010), ce qui paraît un score très élevé pour une méthode naïve.

Cela nous laisse penser que, même si l'inclusion de modèles statistiques par n-grammes de mots, nous permettrait de gagner sur la qualité de la correction en contexte, le gain à espérer est sans doute plus faible que celui estimé ici.

10.4 Étude de la représentation dans le cadre de traitements journaliers en apprentissage supervisé

L'évolution de l'offre en 2010 nous pousse à considérer de nouvelles techniques d'apprentissage pour des traitements réalisés quotidiennement.

La monotonie de la taxonomie d'une journée à l'autre garanti au client une lecture aisée de l'évolution d'une même idée au fil du temps. En terme d'apprentissage, nous ne devons plus découvrir des régularités dans les données présentées pour établir une synthèse mais apprendre au fil des traitements journaliers à reconnaître les lois unissant les données en entrées et les classes de la taxonomie définies avec le client. Notre problématique se situe donc désormais dans le domaine de l'apprentissage supervisé.

Les questionnaires employés reposent dorénavant sur les trois questions NPS que nous avons présentées à la section 9.2.1 page 156, nous donnant ainsi un cadre de travail plus restreint et plus spécifique.

Dans cette section, nous présentons notre stratégie d'apprentissage supervisé reposant sur un proche voisinage puis nous abordons les attributs supplémentaires que nous avons choisi de prendre en compte dans la représentation des textes dans le cadre de questionnaires NPS. Nous montrons notamment l'impact de ces attributs sur la qualité de l'apprentissage réalisé. Nous présentons enfin une comparaison de nos méthodes avec les algorithmes à base de machines à vecteurs supports largement employés dans la classification textuelle.

10.4.1 Apprentissage supervisé par l'algorithme des K-plus proches voisins

De par sa simplicité implémentatoire et surtout sa validité bayésienne, l'algorithme des K-plus proches voisins semble un choix judicieux pour réaliser l'apprentissage supervisé.

Nous nous intéressons notamment à sa version la plus simple : l'algorithme du plus proche voisin classant une donnée inconnue comme la donnée la plus proche connue de l'échantillon d'apprentissage. Cet algorithme correspond donc formellement à l'algorithme des K-plus proches voisins lorsque $K = 1$.

Cet algorithme (cf. section 7.2.1) ne peut produire un résultat que sur des verbatims ne comportant qu'une seule idée. Nous proposons donc d'utiliser l'adaptation de l'algorithme des plus proches voisins présentée à la section 10.2.6 afin de classer des verbatims comportant plusieurs idées. Nous nous appuyons sur notre solution de reconnaissance des verbatims comportant plusieurs idées présentée à la section 10.2.

Lors de la classification d'un verbatim, nous distinguons ainsi deux cas :

- Le verbatim ne comporte qu'une seule idée : l'algorithme du plus proche voisin classique est employé.
- Le verbatim comporte plusieurs idées : l'algorithme modifié des plus proches voisins pour prendre en compte plusieurs idées est employé.

Si l'on tolère la présence de verbatims comportant plusieurs idées dans l'échantillon d'apprentissage, un verbatim détecté comme comportant une seule idée peut

être trouvé en plus proche voisinage avec un verbatim en comportant plusieurs. Dans ce cas, l'algorithme est contraint de le codifier comme comportant plusieurs idées. Nous faisons le choix de privilégier les verbatims ne comportant qu'une seule idée dans l'échantillon d'apprentissage afin d'éviter cet écueil.

10.4.2 Utilisation d'un attribut numérique dans la classification

Nous nous intéressons à présent aux questionnaires de type NPS. Ces derniers présentent trois questions dont la première est une question conduisant à une réponse numérique entière comprise entre 0 et 10 inclus. Cette réponse correspond à une note de satisfaction dénotée score NPS et renseignant sur la loyauté du client envers le service au travers de sa promptitude à le recommander.

Nous montrons dans cette sous-section comment nous proposons d'exploiter ce score NPS afin d'augmenter la précision de notre système de classification.

Définition de classes et coefficients alphanumériques à partir de la note NPS

La question NPS ne fait pas l'objet d'une classification. Cependant, la deuxième question du questionnaire porte directement sur ce score et plus précisément sur le ou les motifs conduisant le client à l'attribuer. Il est alors raisonnable de supposer que la prise en compte du score dans la classification de cette deuxième question peut être utile.

Dans l'article fondateur du NPS (REICHHELD 2003), le score permet de classer les sondés et leurs réponses en trois classes distinctes :

- La classe des détracteurs qui regroupe les sondés ayant mis un score inférieur ou égal à 6.
- La classe des neutres qui regroupe les sondés ayant mis 7 ou 8.
- La classe des promoteurs qui regroupe les sondés ayant mis 9 ou 10.

Pour Reichheld, ces trois classes correspondent aux trois catégories de clients différents en fonction de leur loyauté envers le service qui leur est rendu. Notons que la classe des détracteurs est assez vaste en comparaison des deux autres puisqu'elle

regroupe à elle seule 7 des 11 scores possibles, ne laissant que deux scores pour chacune des deux autres classes restantes. Ce choix est justifié par la volonté de lier la loyauté ainsi mesurée des clients d'un service à la croissance de l'entreprise le rendant. Dans notre étude, nous nous servons du score NPS non pas pour mesurer une croissance mais pour mieux identifier l'opinion du client. Il nous paraît ainsi judicieux de proposer d'autres classifications des sondés reflétant à notre sens les classes d'opinions. Nous proposons ainsi d'étudier les classifications suivantes :

- 0-4/5-6/7-10
- 0-4/5-6/7-8/9-10
- 0-5/6-7/8-10
- 0-6/7-8/9-10

La dernière classification correspond à celle proposée par Reichheld (REICHHELD 2003).

Nous proposons également d'étudier l'utilisation de la note NPS numérique directement sans l'établissement d'une classification.

Soient r_1 et r_2, deux répondants à un questionnaire de satisfaction NPS. Soient nps_1 et nps_2, les deux notes attribuées par ces répondants à la première question du questionnaire et soient v_1 et v_2, les verbatims constituant leurs réponses à la deuxième question du questionnaire NPS.

Afin d'utiliser la note NPS pour la classification des verbatims, il convient de définir une similarité hybride basée sur :

1. la similarité établie par rapprochement des données textuelles des verbatims, notée $Sim_{text}(v_1, v_2)$ (voir section 10.1.2)
2. la similarité entre les classes $c(.)$ établies sur les notes NPS des répondants, notée $Sim_{alpha}(r_1, r_2)$ tel que

$$Sim_{alpha}(r_1, r_2) = \begin{cases} 1 & \Leftrightarrow \quad c(nps_1) = c(nps_2) \\ 0 & \Leftrightarrow \quad c(nps_1) \neq c(nps_2) \end{cases}$$

3. la similarité entre les notes NPS des répondants, notée $Sim_{num}(r_1, r_2)$ telle que

$$Sim_{num}(r_1, r_2) = \frac{\min\{nps_1, nps_2\}}{\max\{nps_1, nps_2\}}$$

La similarité hybride, notée $Sim_{agg}(v_1, v_2, cf_{text}, cf_{alphanum})$, est obtenue par combinaison des similarités textuelles et numériques au moyen des deux coefficients cf_{text} et $cf_{alphanum}$. Ils sont appliqués aux deux similarités selon les calculs suivants :

$$Sim_{alphanum}(v_1, v_2) = \begin{cases} Sim_{num}(r_1, r_2) & \text{ssi aucune classification n'est employée} \\ Sim_{alpha}(r_1, r_2) & \text{ssi une classification est employée} \end{cases}$$

$$cf_{text} + cf_{alphanum} = 1$$

$$Sim_{agg}(v_1, v_2, cf_{text}, cf_{alphanum}) = Sim_{text}(v_1, v_2) * cf_{text} + Sim_{alphanum}(v_1, v_2) * cf_{alphanum}$$

Optimisation des coefficients textuel et alphanumérique

Nous devons réaliser des évaluations pour déterminer la meilleur valeur pour chacun des deux coefficients.

A cette fin, nous disposons :

- d'un ensemble de verbatims d'apprentissage pour chaque idée de la synthèse. Ces verbatims ont été étiquetés au préalable selon les idées exprimées.
- d'un ensemble de test permettant l'évaluation qualitative du paramétrage testé. Ces verbatims ont eux aussi été étiquetés au préalable selon les idées exprimées mais l'information ne sera pas disponible pour le système apprenant. Elle servira uniquement pour l'évaluation.

L'ensemble d'apprentissage que nous utilisons dispose de 1815 verbatims répartis dans 47 classes. Les exemples de cet ensemble sont tous étiquetés soit par la note NPS attribuée par le répondant, soit par la classe correspondante à la note attribuée selon la classification NPS envisagée. Ces verbatims ont par ailleurs été classés au préalable et validés par les codificateurs. L'exemple B.1 donne quelques illustrations des éléments de l'ensemble d'apprentissage.

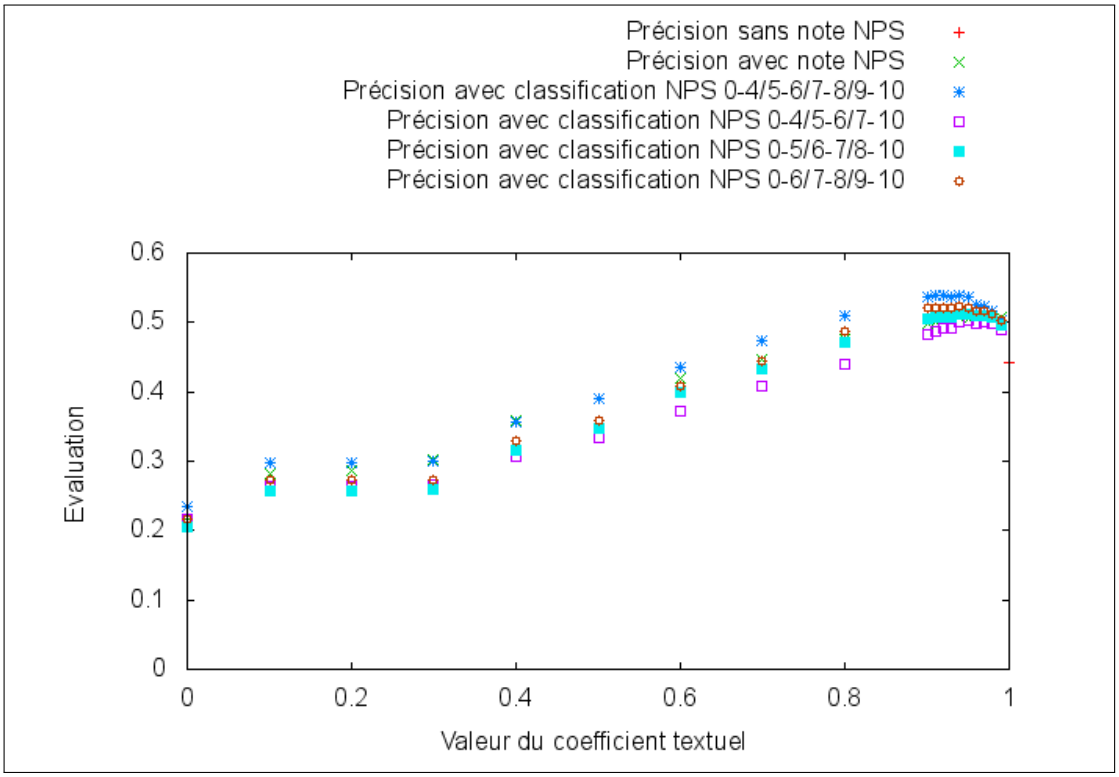


FIGURE 10.9 – Précision en fonction de la valeur des coefficients des attributs textuels et alphanumériques pour différentes classifications NPS sur l'ensemble d'optimisation. La précision du système de classification croît avec la valeur du coefficient textuel jusque 0.95 puis décroît légèrement et ce quelle que soit la classification NPS choisie. Nous notons que la classification NPS 0-4/5-6/7-8/9-10 obtient la meilleure précision dans chaque cas. Son maximum est atteint entre 0.9 et 0.95.

L'ensemble de test, quant à lui, comprend 545 verbatims à répartir dans les 47 classes à l'issue de l'apprentissage. La classification de ces verbatims dans ces 47 classes a été réalisée préalablement et surtout validée humainement par nos codificateurs. L'exemple B.2 donne quelques illustrations des éléments de l'ensemble de test.

Les classifications préalables, validés par les codificateurs et que l'on peut supposer raisonnablement fiables, constituent nos référentiels d'apprentissage et d'évaluation selon lesquels nous évaluons les prédictions de notre système.

Les résultats obtenus sur l'ensemble de test après apprentissage sont donnés par les figures 10.9, 10.10 et 10.11 pour les différentes classifications NPS testées.

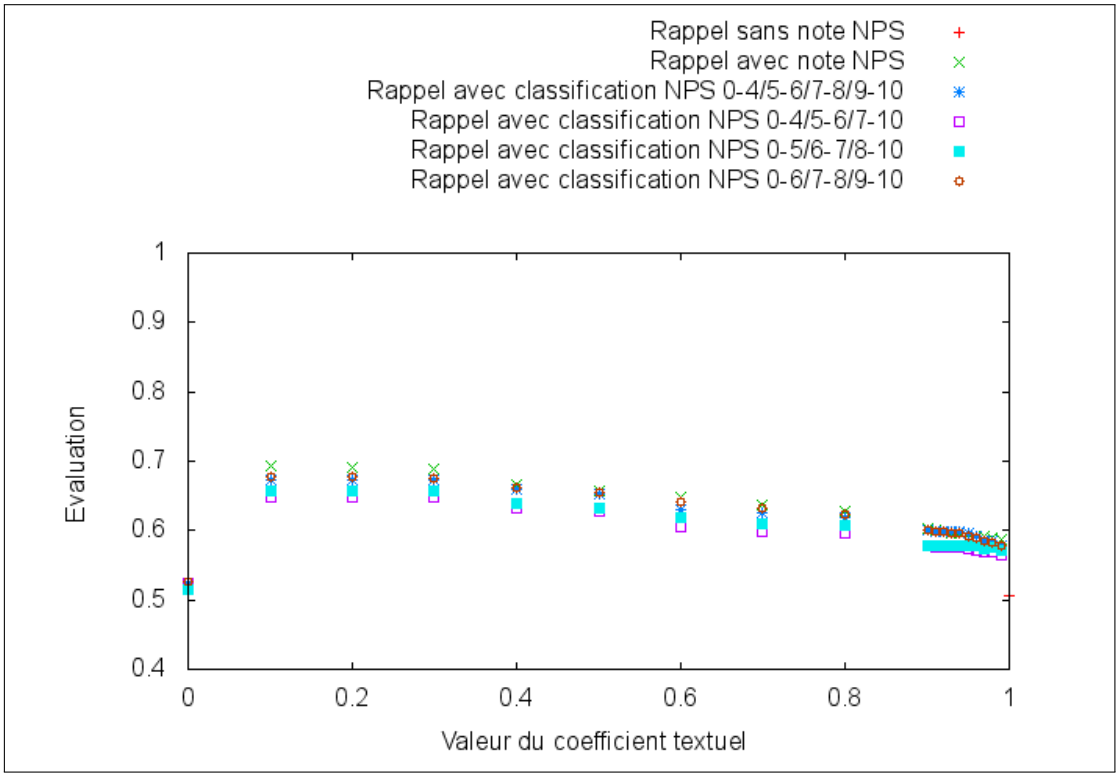


FIGURE 10.10 – Rappel en fonction de la valeur des coefficients des attributs textuels et alphanumériques pour différentes classifications NPS sur l'ensemble d'optimisation. Le rappel du système de classification décroît avec la valeur du coefficient textuel entre 0.1 et 1.0. Sa valeur la plus faible est atteinte pour une valeur du coefficient textuel à 0. Aucune des classifications NPS testées ne se démarque clairement sur le rappel.

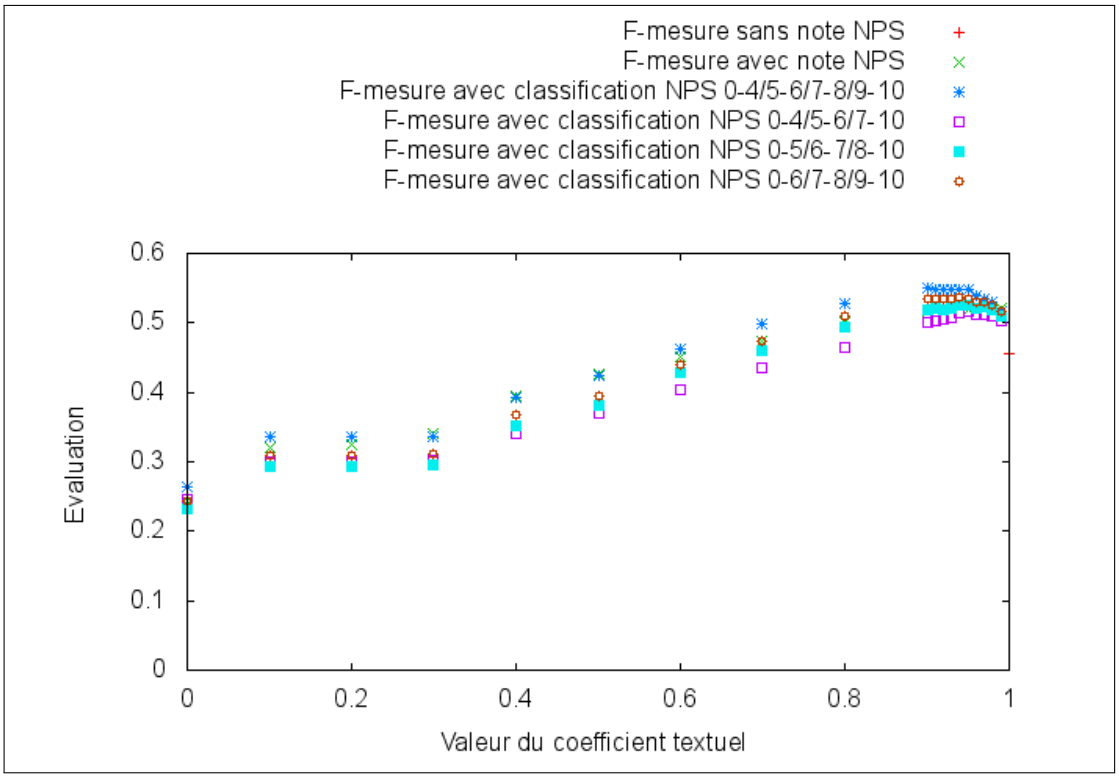


FIGURE 10.11 – F-mesure ($\beta = 0.5$) en fonction de la valeur des coefficients des attributs textuels et alphanumériques pour différentes classifications NPS sur l'ensemble d'optimisation. La f-mesure du système de classification croît avec la valeur du coefficient textuel jusque 0.95 puis décroît légèrement et ce quelle que soit la classification NPS choisie. Nous notons que la classification NPS 0-4/5-6/7-8/9-10 obtient la meilleure f-mesure à chaque cas. Son maximum est atteint entre 0.9 et 0.95. Son minimum est atteint pour une valeur de 0 du coefficient textuel.

Les résultats montrent que la classification NPS 0-4/5-6/7-8/9-10 obtient les meilleurs résultats que ce soit en précision ou en f-mesure avec une valeur du coefficient textuel entre 0.9 et 0.95. Ils montrent également l'importance de la similarité textuelle puisque les pires résultats sont obtenus en n'en tenant pas compte.

Optimisation du rayon de voisinage à considérer

Le rayon de voisinage, noté R_N, a été jusqu'ici supposé idéal pour une valeur de 0.6. Le choix de cette valeur ne s'appuie pas sur une optimisation rigoureuse mais uniquement sur une supposition raisonnable. Nous proposons donc dans cette sous-section d'optimiser la valeur du rayon de voisinage.

Nous utilisons à cette fin les deux mêmes ensembles d'apprentissage et de test que ceux utilisés précédemment pour l'optimisation des coefficients textuels et alphanumériques.

Les résultats obtenus sur l'ensemble de test après apprentissage sont donnés par les figures 10.12, 10.13 et 10.14 pour les différentes classifications NPS testées.

Les résultats montrent que la classification NPS 0-4/5-6/7-8/9-10 obtient les meilleurs résultats que ce soit en précision ou en f-mesure avec une valeur du rayon de voisinage de 0.7 ou 0.8. Ils montrent également l'importance de la similarité basée sur la note NPS puisque les pires résultats sont obtenus en n'en tenant pas compte. Nous présentons par la suite de nouvelles évaluations visant à départager les deux valeurs de paramètres trouvées ici.

Évaluations

Afin de valider les deux optimisations faites précédemment et surtout valider que les valeurs identifiées pour les paramètres sont corrects quel que soit le traitement, il convient de réaliser un nouveau test et de vérifier que l'optimum reste inchangé.

Pour cela, nous conservons le même ensemble d'apprentissage que celui utilisé précédemment mais prenons en revanche deux nouveaux ensembles de test constitués respectivement de 561 et 5213 verbatims toujours à classer dans les 47 classes à l'issue de l'apprentissage. La classification de ces verbatims dans ces 47 classes est

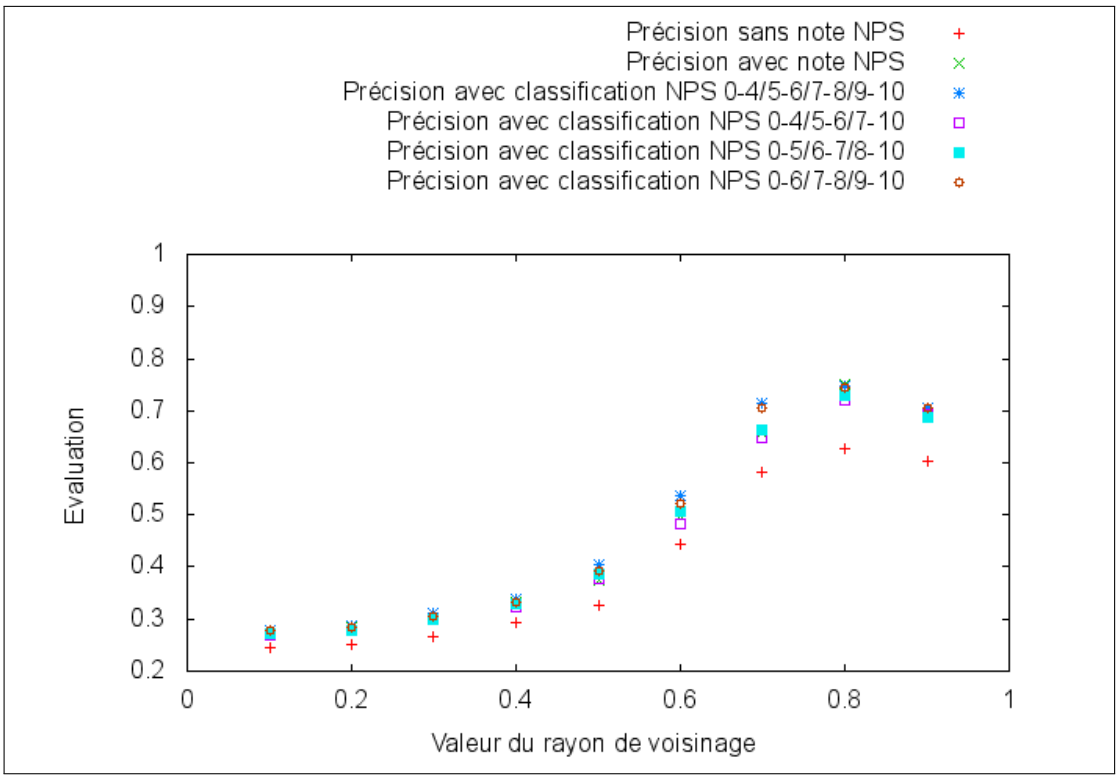


FIGURE 10.12 – Précision en fonction de la valeur du rayon de voisinage pour différentes classifications NPS sur l'ensemble d'optimisation. La précision du système de classification croît avec la valeur du rayon de voisinage jusque 0.8 puis décroît légèrement et ce quelle que soit la classification NPS choisie. Nous notons que la classification NPS 0-4/5-6/7-8/9-10 obtient la meilleure précision à chaque cas. Son maximum est atteint pour un rayon de voisinage de 0.8.

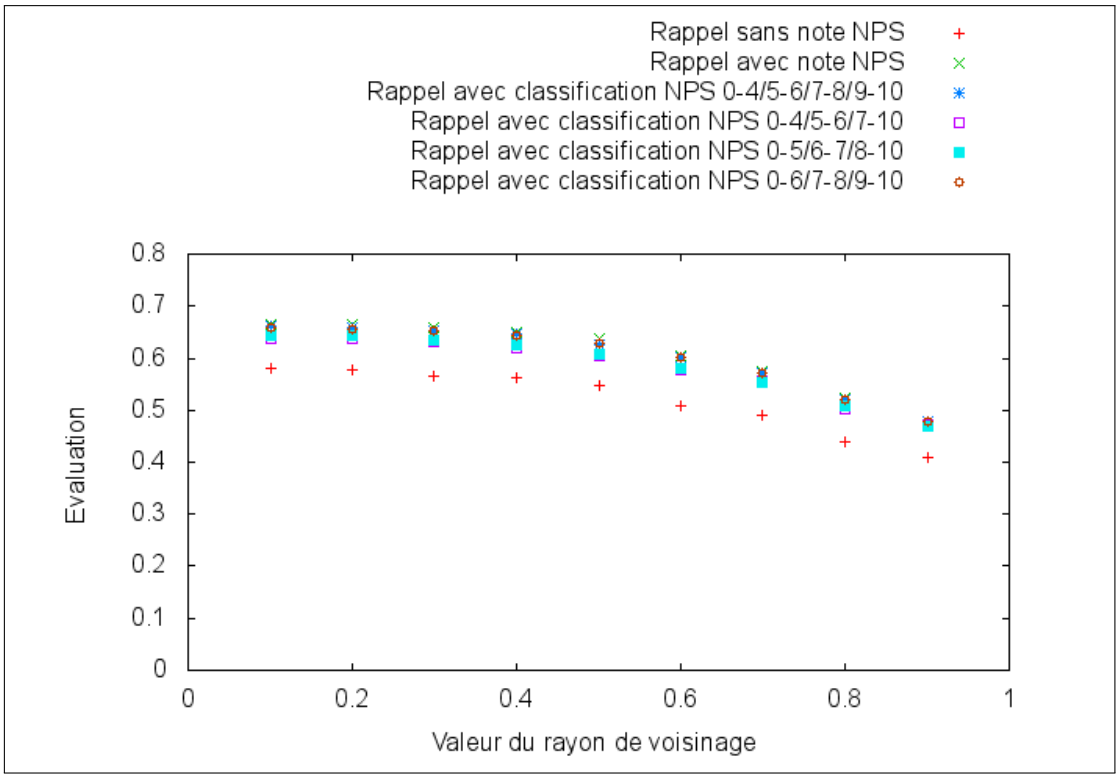


FIGURE 10.13 – Rappel en fonction de la valeur du rayon de voisinage pour différentes classifications NPS sur l'ensemble d'optimisation. Le rappel du système de classification décroît avec la valeur du rayon de voisinage entre 0.1 et 0.9. Aucune des classifications NPS testées ne se démarque clairement sur le rappel si ce n'est le cas ne tenant pas compte de la note NPS qui obtient toujours le pire rappel.

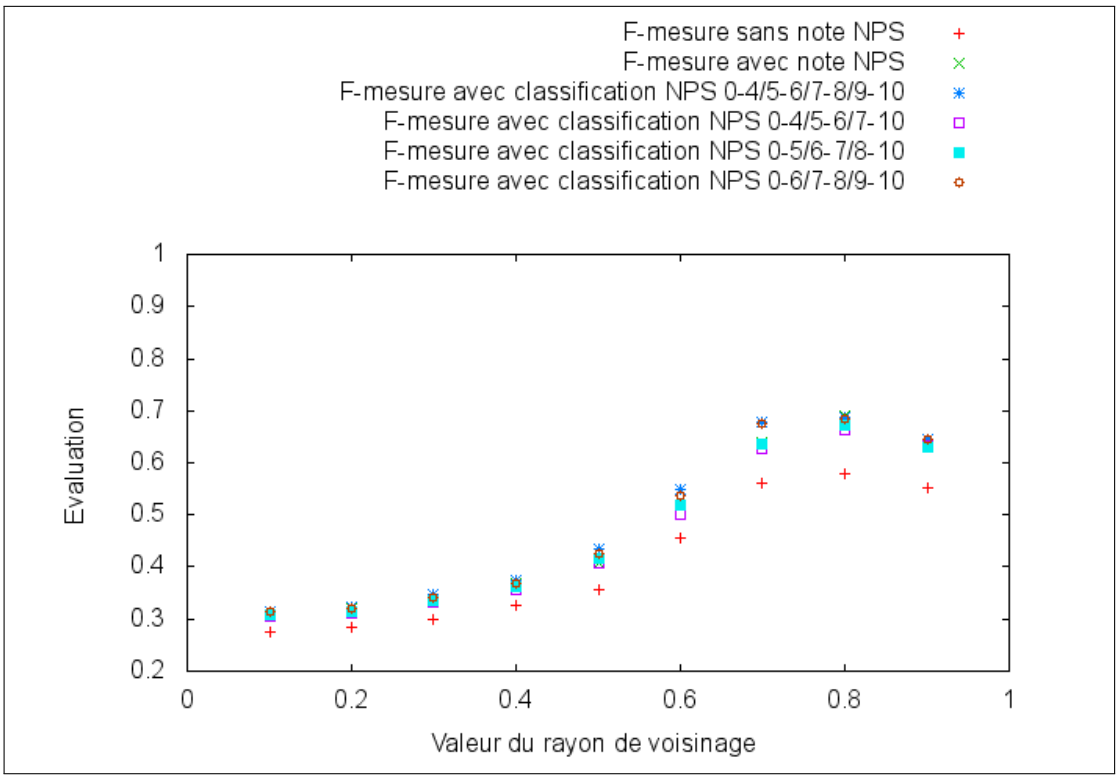


FIGURE 10.14 – F-mesure ($\beta = 0.5$) en fonction de la valeur du rayon de voisinage pour différentes classifications NPS sur l'ensemble d'optimisation. La f-mesure du système de classification croît avec la valeur du rayon de voisinage jusque 0.8 puis décroît légèrement et ce quelle que soit la classification NPS choisie. Nous notons que la classification NPS 0-4/5-6/7-8/9-10 obtient la meilleure f-mesure à chaque cas. Son maximum est atteint pour un rayon de voisinage de 0.7 ou 0.8.

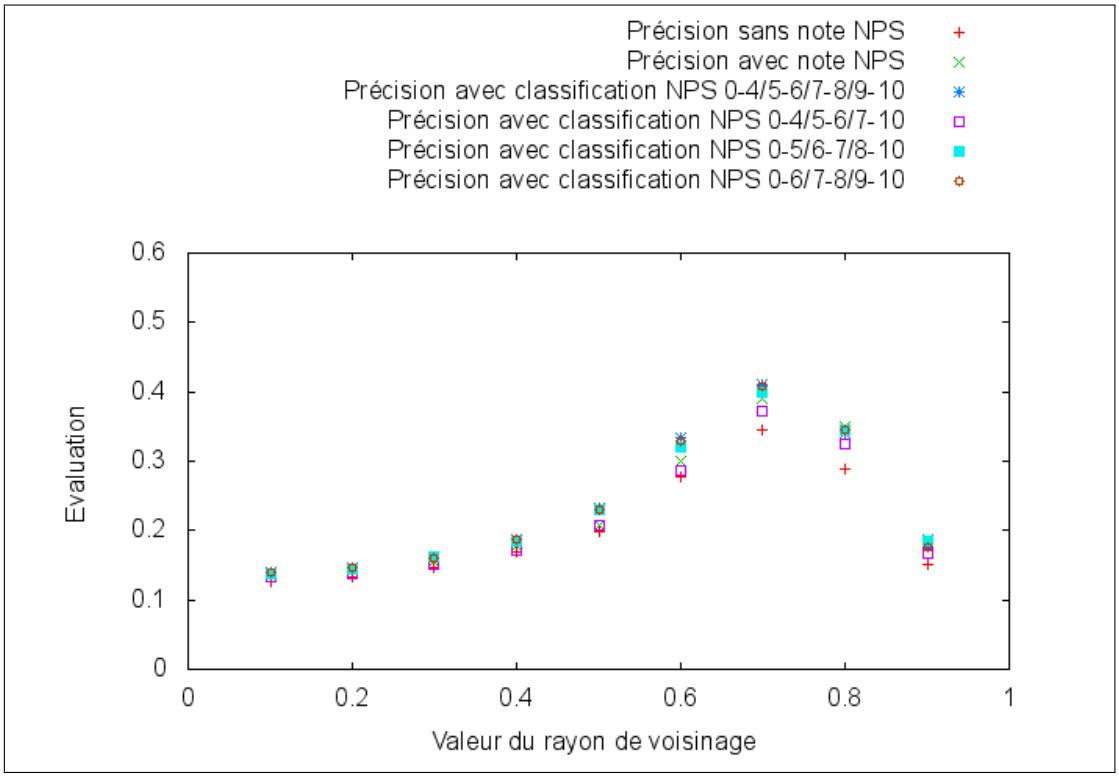


FIGURE 10.15 – Précision en fonction de la valeur du rayon de voisinage pour différentes classifications NPS sur le 1er ensemble d'évaluation. La précision du système de classification croît avec la valeur du rayon de voisinage jusque 0.7 puis chute rapidement et ce quelle que soit la classification NPS choisie. Nous notons qu'il n'y plus cette fois de classification nettement meilleure que les autres mais que la non-prise en compte du NPS donne les pires résultats. Les maxima sont atteints pour un rayon de voisinage de 0.7.

toujours réalisée préalablement et validée humainement par nos codificateurs. Les exemples B.3 et B.4 donnent quelques illustrations des nouveaux éléments des ensembles de test.

Les résultats obtenus sur l'ensemble d'évaluation après apprentissage sont donnés par les figures 10.15, 10.16 et 10.17 pour les différentes classifications NPS testées.

Les résultats ne montrent plus une classification NPS obtenant de meilleurs résultats que les autres mais confirment l'importance de la similarité basée sur la note NPS puisque les pires résultats sont encore obtenus en n'en tenant pas compte. Les meilleurs résultats sont également clairement obtenus pour un rayon de voisinage de 0.7.

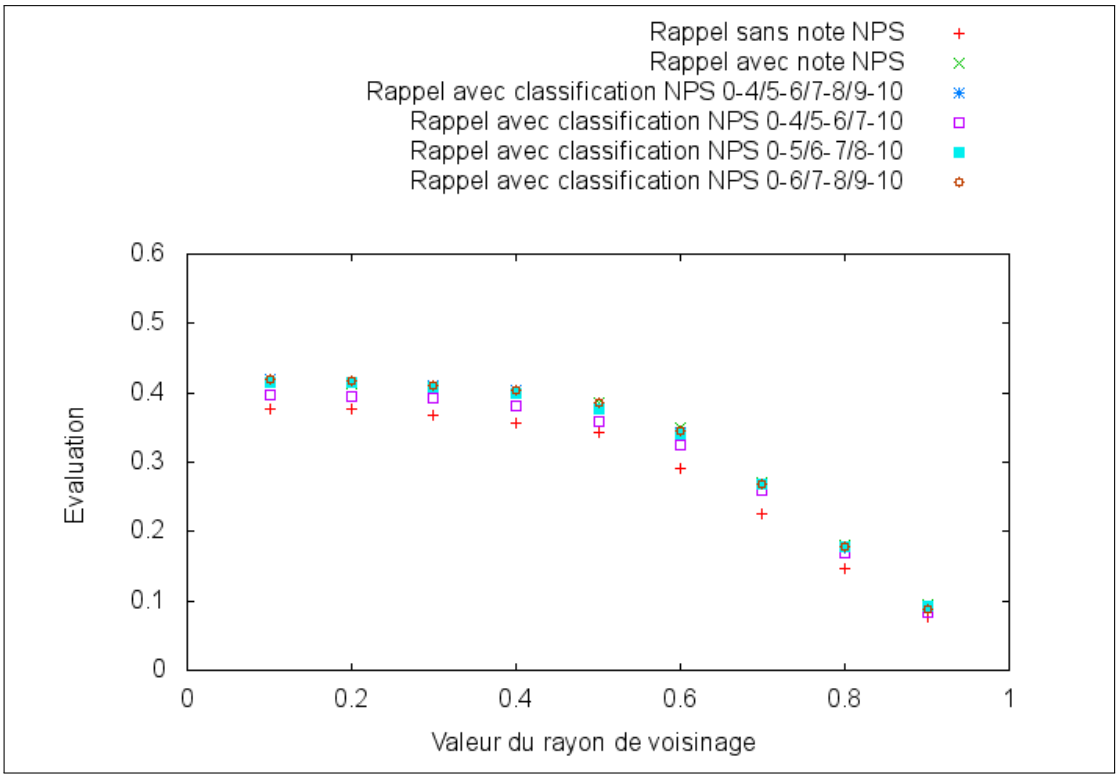


FIGURE 10.16 – Rappel en fonction de la valeur du rayon de voisinage pour différentes classifications NPS sur le 1er ensemble d'évaluation. Le rappel du système de classification décroît avec la valeur du rayon de voisinage entre 0.1 et 0.9. Aucune des classifications NPS testées ne se démarque clairement sur le rappel si ce n'est le cas ne tenant pas compte de la note NPS qui obtient toujours le pire rappel.

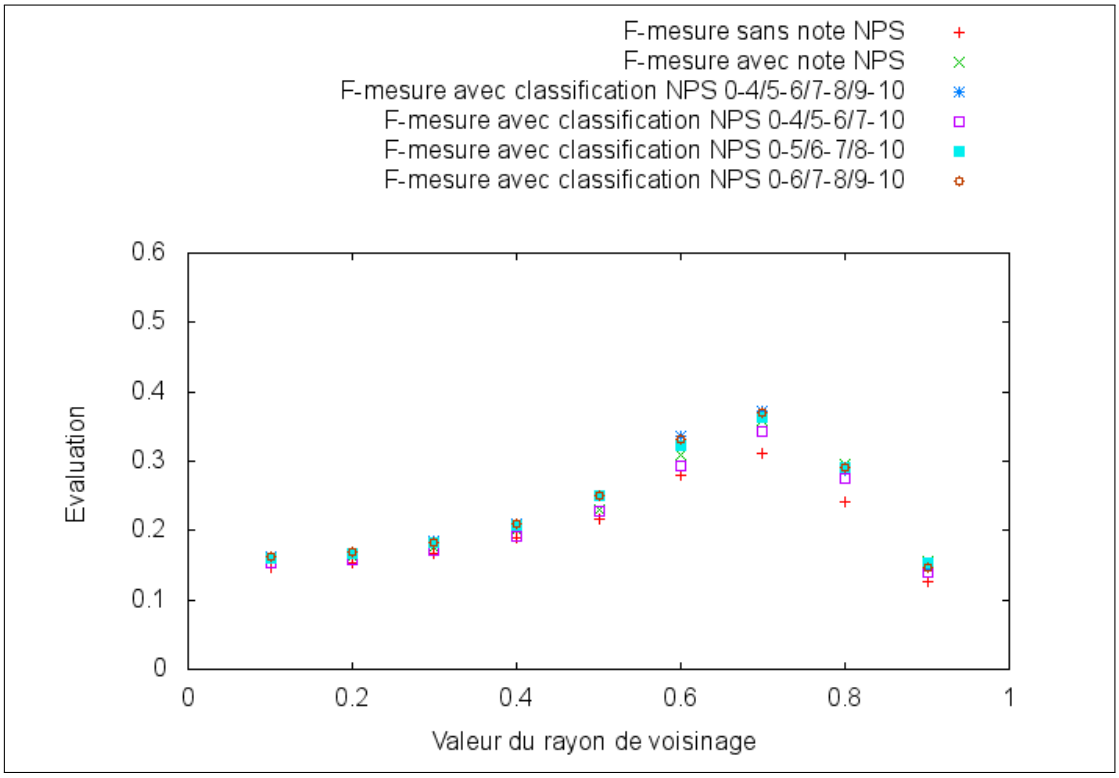


FIGURE 10.17 – F-mesure ($\beta = 0.5$) en fonction de la valeur du rayon de voisinage pour différentes classifications NPS sur le 1er ensemble d'évaluation. La f-mesure du système de classification croît avec la valeur du rayon de voisinage jusque 0.7 puis chute rapidement et ce quelle que soit la classification NPS choisie. Nous notons qu'il n'y plus cette fois de classification nettement meilleure que les autres mais que la non-prise en compte du NPS donne les pires résultats. Les maxima sont atteints pour un rayon de voisinage de 0.7.

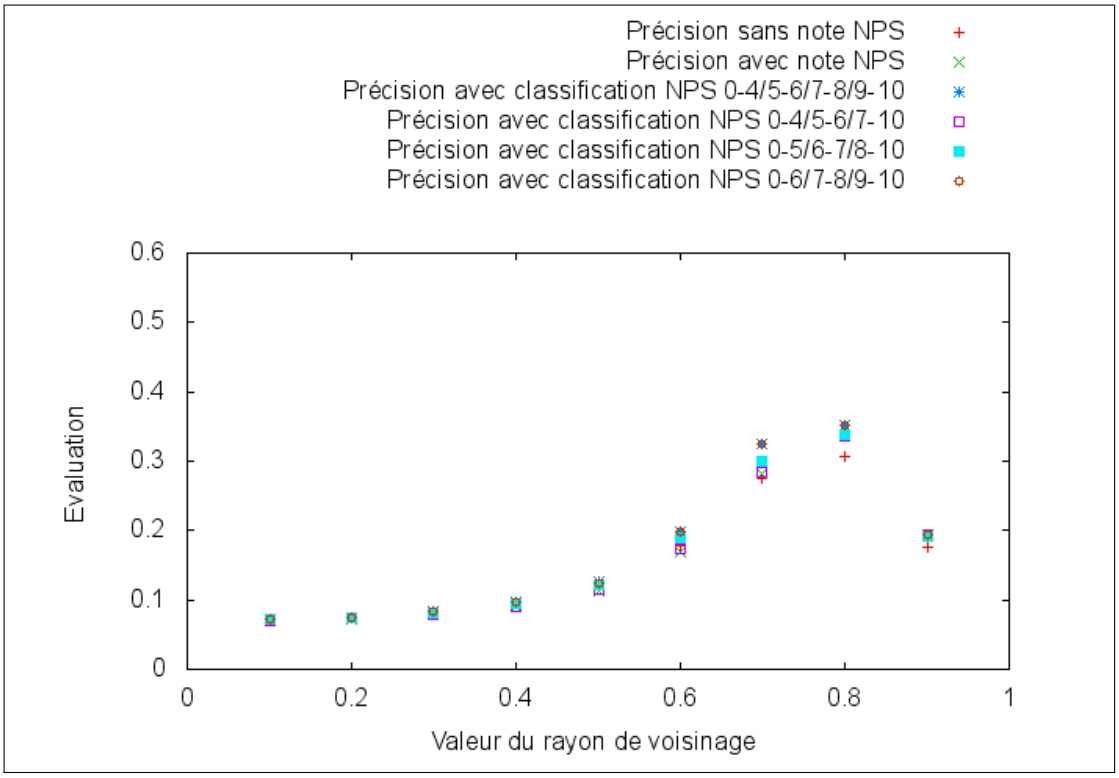


FIGURE 10.18 – Précision en fonction de la valeur du rayon de voisinage pour différentes classifications NPS sur le 2e ensemble d'évaluation. La précision du système de classification croît avec la valeur du rayon de voisinage jusque 0.8 puis chute rapidement et ce quelle que soit la classification NPS choisie. Nous notons qu'il n'y plus cette fois de classification nettement meilleure que les autres mais que la non-prise en compte du NPS donne les pires résultats. Les maxima sont atteints pour un rayon de voisinage de 0.8.

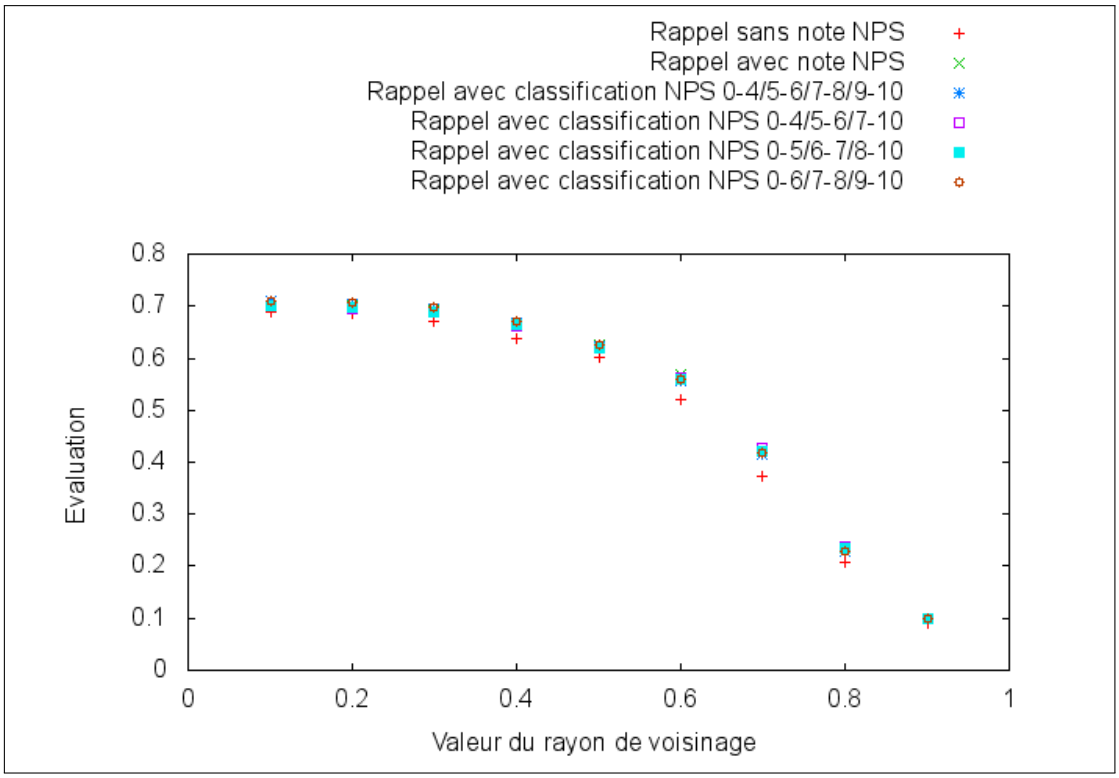


FIGURE 10.19 – Rappel en fonction de la valeur du rayon de voisinage pour différentes classifications NPS sur le 2e ensemble d'évaluation. Le rappel du système de classification décroît avec la valeur du rayon de voisinage entre 0.1 et 0.9. Aucune des classifications NPS testées ne se démarque clairement sur le rappel si ce n'est le cas ne tenant pas compte de la note NPS qui obtient toujours le pire rappel.

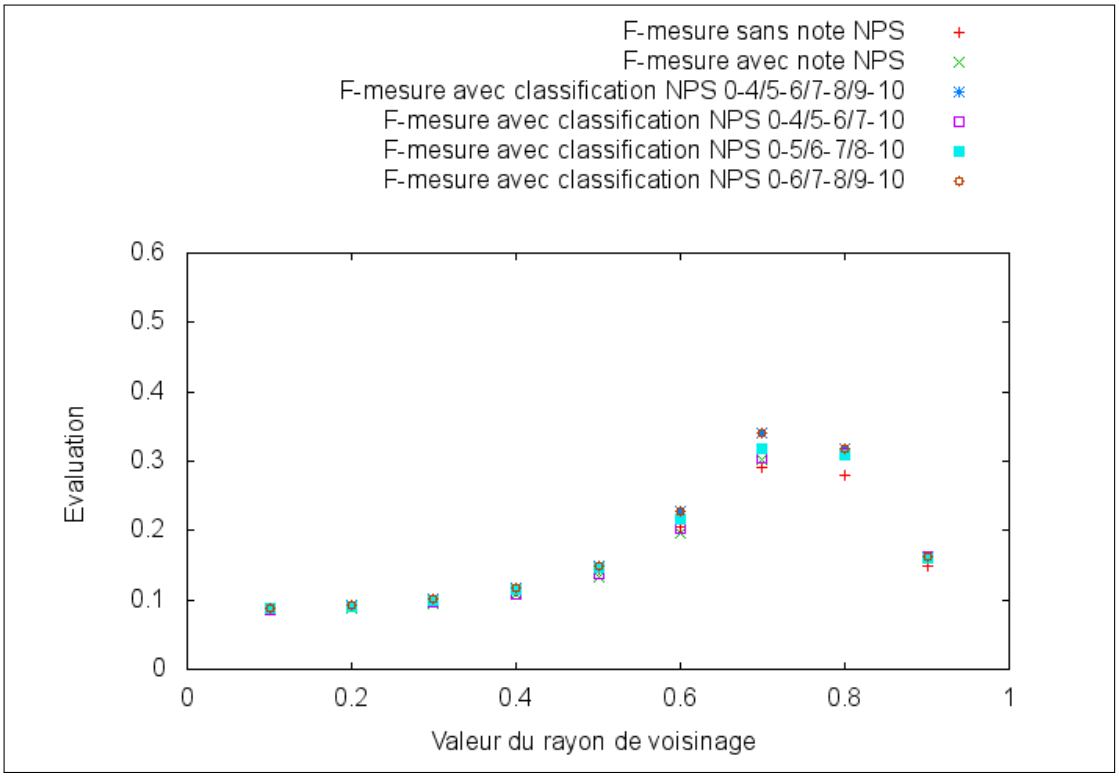


FIGURE 10.20 – F-mesure ($\beta = 0.5$) en fonction de la valeur du rayon de voisinage pour différentes classifications NPS sur le 2e ensemble d'évaluation. La f-mesure du système de classification croît avec la valeur du rayon de voisinage jusque 0.7 puis chute rapidement et ce quelle que soit la classification NPS choisie. Nous notons qu'il n'y plus cette fois de classification nettement meilleur que les autres mais que la non-prise en compte du NPS donne les pires résultats. Les maxima sont atteints pour un rayon de voisinage de 0.7.

Les résultats obtenus sur l'ensemble d'évaluation après apprentissage sont donnés par les figures 10.18, 10.19 et 10.20 pour les différentes classifications NPS testées.

Les résultats ne montrent plus une classification NPS conduisant à de meilleurs résultats que les autres mais confirment l'importance de la similarité basée sur la note NPS puisque les pire résultats sont encore obtenus en n'en tenant pas compte. Les meilleurs résultats sont également clairement obtenus pour un rayon de voisinage de 0.7.

Conclusion des optimisations et évaluations

Les optimisations et évaluations présentées dans cette sous-section nous ont permis d'identifier les paramètres et réglages suivants comme empiriquement optimaux :

- utilisation de la classification NPS 0-4/5-6/7-8/9-10.
- réglage du rayon de voisinage à une valeur de 0.7.
- réglage du coefficient textuel à une valeur de 0.9.
- réglage du coefficient alphanumérique à une valeur de 0.1.

Ils ont par ailleurs mis en évidence l'intérêt de la prise en compte de la note NPS puisque que les pires résultats sont obtenus lorsque l'on n'en tient pas compte.

10.4.3 Comparaison de l'apprentissage supervisé avec l'algorithme SVM

L'algorithme SVM présenté à la section 7.2.2 est un algorithme d'apprentissage supervisé réputé en classification. Sa relative complexité de mise en place en comparaison des K-plus proches voisins nous pousse à ne le considérer que dans un second temps.

L'avantage de SVM, contrairement aux K-plus proches voisins, réside dans sa capacité à trouver des relations non linéaires sur les attributs de l'espace d'entrée tout en conservant une simplicité calculatoire grâce à l'emploi de fonctions noyaux.

Il nous paraît donc judicieux de comparer notre approche basée sur les K-plus proches voisins avec celle basée sur les SVM.

Nous souhaitons bien sûr utiliser les mêmes représentations de textes avec les SVM qu'avec notre approche. Nous ne devons pour cela pas fournir à SVM des dimensions reflétant les mots présents dans les messages à classer mais plutôt des dimensions traduisant l'éloignement du message à classer avec d'autres messages que nous appellerons messages de référence.

A cette fin, nous disposons :

- d'un ensemble de verbatims de référence pour chaque idée de la synthèse.
- d'un ensemble de verbatims d'apprentissage pour chaque idée de la synthèse. Ces verbatims ont par ailleurs été étiquetés au préalable selon les idées exprimées.
- d'un ensemble de test permettant l'évaluation qualitative du paramétrage testé. Ces verbatims ont eux aussi été étiquetés au préalable selon les idées exprimées mais l'information ne sera pas disponible pour le système apprenant. Elle servira uniquement pour l'évaluation.

L'ensemble de référence que nous utilisons dispose de 1815 verbatims répartis dans 47 classes. Les exemples cet ensemble sont tous étiquetés soit par la note NPS attribuée par le répondant, soit par la classe correspondante à la note attribuée selon la classification NPS envisagée. Ces verbatims ont par ailleurs été classés au préalable et validés par les codificateurs. Notons qu'il s'agit simplement des verbatims utilisés préalablement pour l'apprentissage selon notre méthode basée sur les K-plus proches voisins.

L'ensemble d'apprentissage que nous utilisons dispose de 5593 verbatims ne comportant tous qu'une seule des 47 idées exprimées par les 47 classes. Ces verbatims correspondant à l'extraction de l'ensemble des verbatims ne comportant qu'une seule idée traités sur 9 mois, la répartition des verbatims d'entraînement par idée correspond à la proportion constatée en moyenne dans les dits traitements. Cette répartition va ainsi de près de 700 verbatims d'entraînement pour l'idée la plus exprimée à un seul verbatim pour la moins exprimée. Ce constat nous permet de souligner une difficulté importante dans notre classification qu'est l'importante hétérogénéité des données disponibles d'une classe à l'autre. Ainsi, certaines classes disposent de beaucoup de données et sont donc plus faciles à apprendre tandis que d'autres disposent de très peu de données, ce qui rend leur apprentissage très complexe. La ré-

partition précise des verbatims d'apprentissage par idée est donnée dans les tableaux 10.20 et 10.21.

Les figures 10.21, 10.22 et 10.23 donnent les précisions, rappels et f-mesures obtenus sur les 10 idées les plus représentées dans l'ensemble de test que sont :

1. Bonne qualité des produits
2. Bon accueil
3. Bon produits
4. Large choix
5. Bon rapport qualité prix
6. Prix abordable
7. Disponibilité des vendeuses
8. Qualité des conseils
9. Gentillesse des vendeuses
10. Magasin agréable

La précision des SVM est plus importante sur les 4 idées les plus importantes que celle obtenue avec notre algorithme des K-plus proches voisins quel que soit le seuil choisi pour SVM. Pour les 6 idées suivantes, la meilleure précision est partagée entre SVM et notre algorithme basé sur les K-plus proches voisins.

Le rappel est globalement meilleur sur notre algorithme basé sur les K-plus proches voisins que sur les SVM. Seules certaines valeurs de seuils de SVM des idées « bon accueil » et « gentillesse des vendeuses » sont meilleures que celles obtenues avec notre algorithme de proche voisinage.

La F-mesure est globalement meilleure avec les SVM qu'avec notre algorithme de proche voisinage pour les 3 idées les plus représentées mais globalement meilleure avec notre algorithme de proche voisinage pour les suivantes exception faite de la dixième.

10.4.4 Conclusion sur l'apprentissage supervisé

Nous concluons de ces expériences que notre solution de classification basée sur du proche voisinage se révèle pertinente par rapport à un algorithme tel que SVM

Titre de l'idée	# verbatims dans l'ensemble d'apprentissage	% verbatims dans l'ensemble d'apprentissage
Bon accueil / accueil souriant	692	12,37
Bonne qualité des produits	563	10,07
Satisfaite	504	9,01
Gentillesse / professionnalisme des vendeuses	501	8,96
Bons / beaux produits	474	8,47
Bon rapport qualité / prix	367	6,56
Large choix	360	6,44
Manque de choix	225	4,02
Qualité des conseils	129	2,31
Produits du catalogue non présents en boutique	128	2,29
gestion des caisses (temps d'attente...)	118	2,11
Personnel pas aimable	100	1,79
Bons services (échanges....)	97	1,73
Prix élevés	96	1,72
Disponibilité des vendeuses	93	1,66
Magasin agréable / lumineux	88	1,57
Organisation / agencement à revoir ou améliorer	88	1,57
Apprécie la marque	87	1,56
Accueil à améliorer	70	1,25
produits non disponibles	62	1,11
Bonne organisation / agencement	59	1,05
possibilité de toucher les produits	58	1,04

TABLE 10.20 – Répartition par idée des verbatims d'apprentissage utilisés avec l'algorithme SVM (1re partie). Nous notons une très grande hétérogénéité entre les volumes de données disponibles d'une idée à l'autre.

Titre de l'idée	**# verbatims dans l'ensemble d'apprentissage**	**% verbatims dans l'ensemble d'apprentissage**
Qualité insatisfaisante	53	0,95
Produits insatisfaisants	51	0,91
prix abordable	50	0,89
Manque de disponibilité	48	0,86
offres commerciales intéressantes	45	0,80
Écart de prix / d'offres entre la boutique et internet	44	0,79
Disponibilité des produits	39	0,70
Conseils / renseignements à améliorer	37	0,66
Service insatisfaisant	37	0,66
Collection à améliorer / pas à la mode	31	0,55
Bonne implantation du magasin	29	0,52
Petite taille de la boutique	28	0,50
Manque ou absence de produits de décoration / puériculture	26	0,46
Offres commerciales non attractives	23	0,41
Offres commerciales non lisibles	20	0,36
Bonne présentation des produits	17	0,30
Magasin pas agréable	17	0,30
carte de fidélité non satisfaisante	14	0,25
Magasin spacieux	9	0,16
renouvellement de gamme fréquent	4	0,07
Pas assez de cabines d'essayage	4	0,07
Magasin loin	4	0,07
Service rapide (caisse)	2	0,04
Coin enfant	1	0,02
Coin enfant pas satisfaisant	1	0,02
Total	5593	

TABLE 10.21 – Répartition par idée des verbatims d'apprentissage utilisés avec l'algorithme SVM (2e partie). Nous notons une très grande hétérogénéité entre les volumes de données disponibles d'une idée à l'autre.

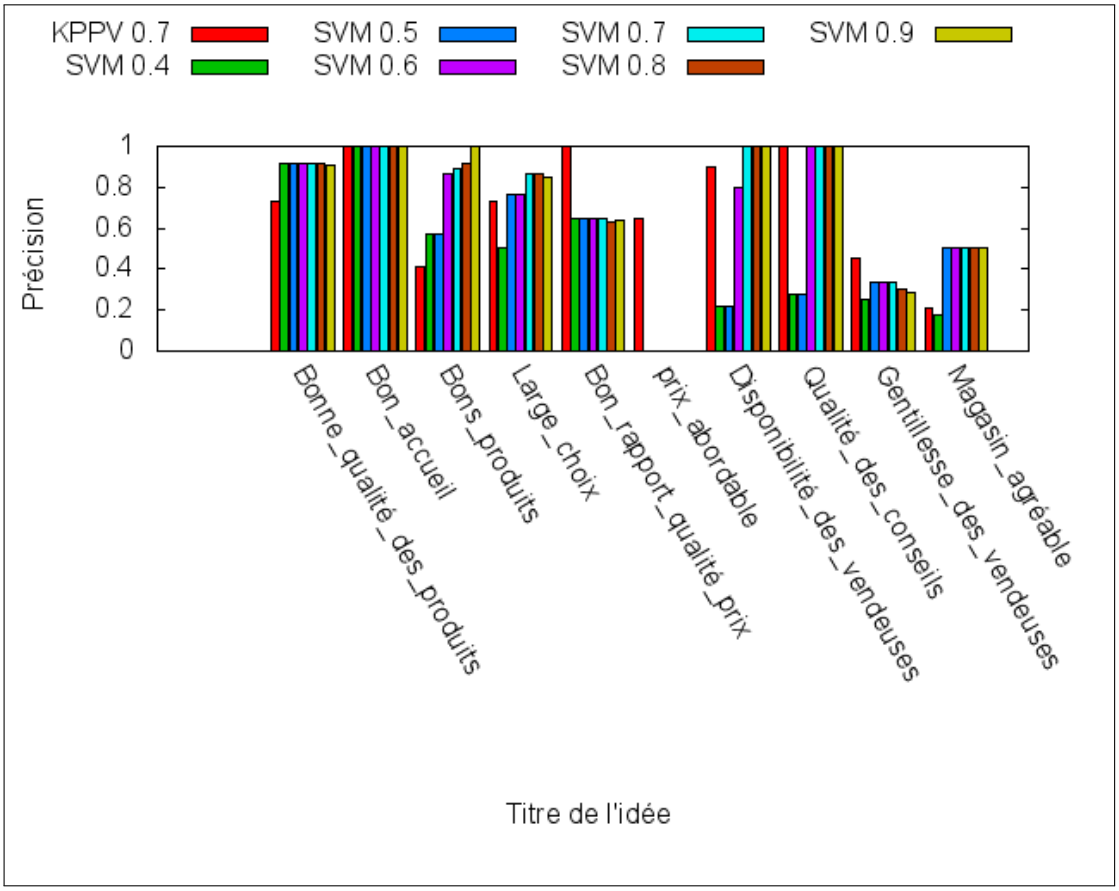


FIGURE 10.21 – Précision sur les 10 idées les plus représentées dans l'ensemble de test pour différents seuils de classification par l'algorithme SVM comparée à la précision sur chaque idée pour notre algorithme basé sur les K-plus proches voisins. Les 10 idées les plus représentées sont classées dans le graphique de la plus importante à gauche à la moins importante à droite.

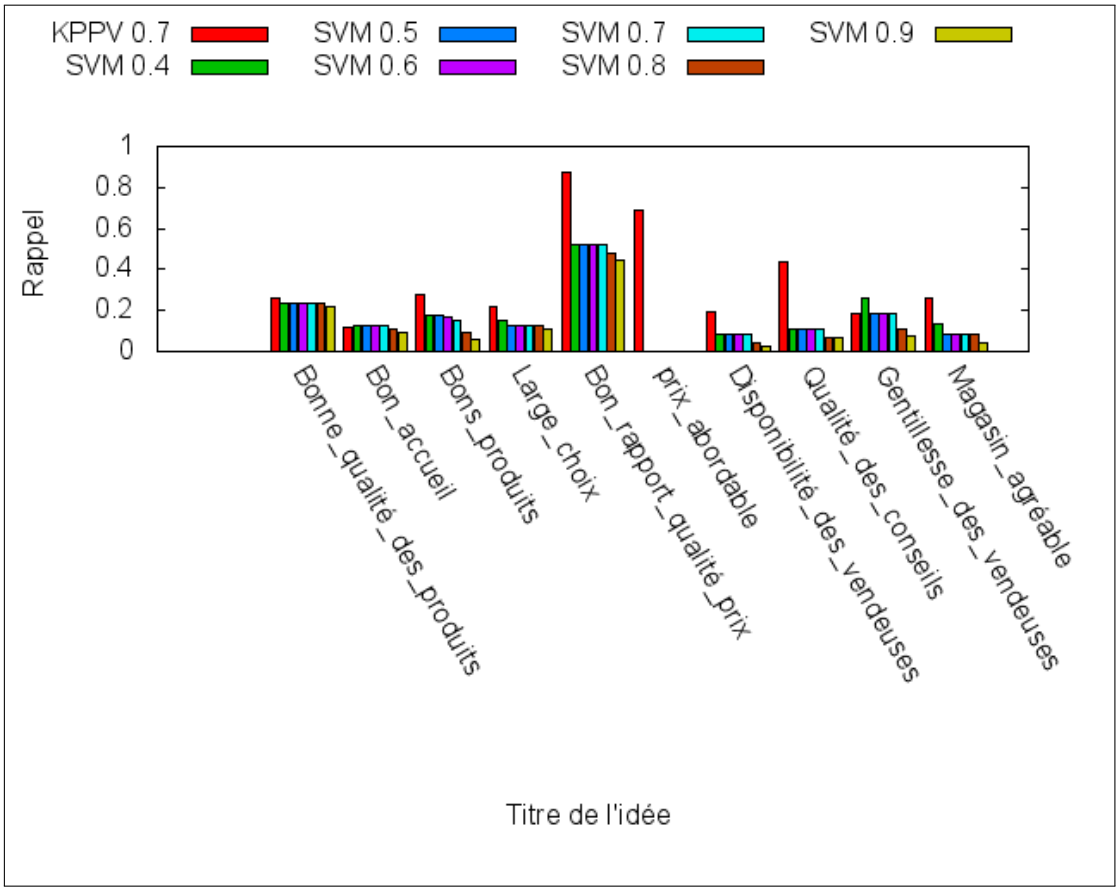


FIGURE 10.22 – Rappel sur les 10 idées les plus représentées dans l'ensemble de test pour différents seuils de classification par l'algorithme SVM comparé au rappel sur chaque idée pour notre algorithme basé sur les K-plus proches voisins. Les 10 idées les plus représentées sont classées dans le graphique de la plus importante à gauche à la moins importante à droite.

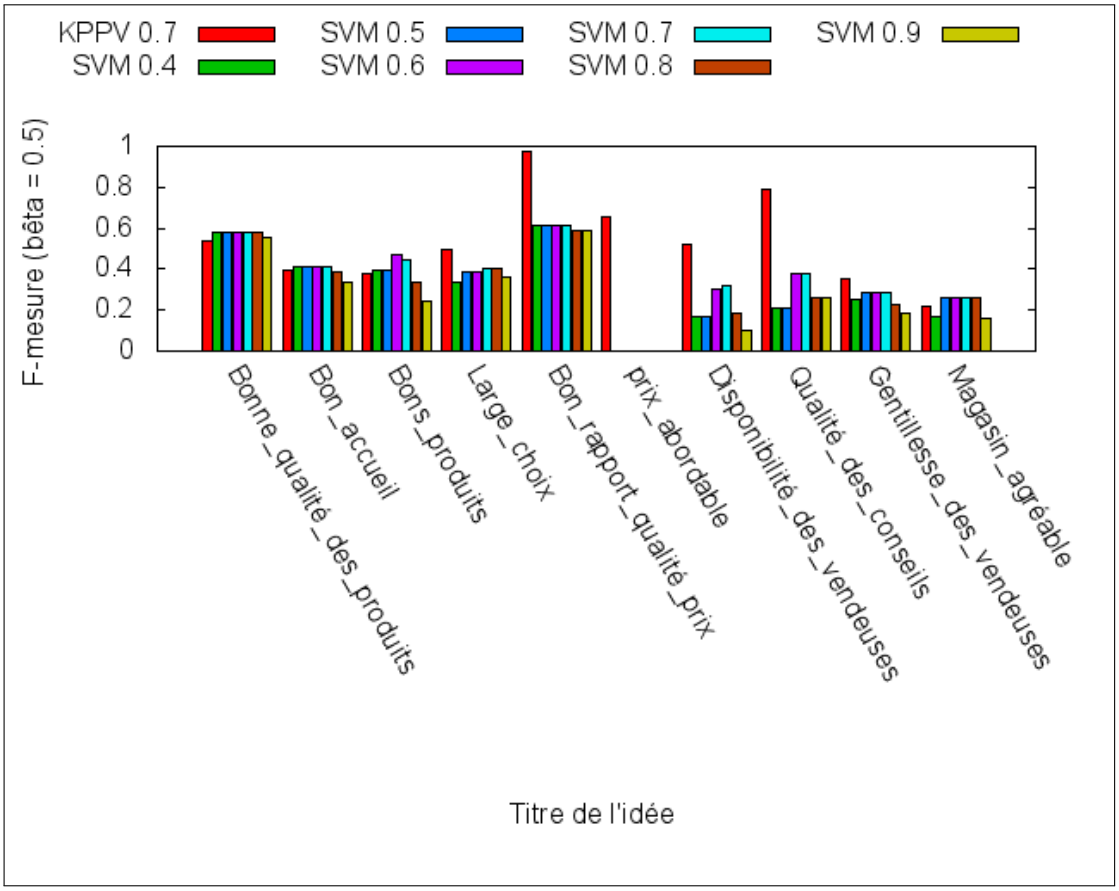


FIGURE 10.23 – F-mesure ($\beta = 0.5$) sur les 10 idées les plus représentées dans l'ensemble de test pour différents seuils de classification par l'algorithme SVM comparée à la f-mesure sur chaque idée pour notre algorithme basé sur les K-plus proches voisins. Les 10 idées les plus représentées sont classées dans le graphique de la plus importante à gauche à la moins importante à droite.

lorsque peu d'exemples d'apprentissage sont disponibles. En revanche, lorsque SVM dispose de suffisamment d'exemples pour construire un modèle de classification robuste, il est alors meilleur que notre algorithme.

Ce constat peut provenir de la capacité de SVM à exploiter les différences entre exemples et contre-exemples pour rendre son modèle plus fiable contrairement au proche voisinage qui ne se base que sur les exemples. Cet atout lorsque les informations sont disponibles est probablement aussi son point faible lorsque les exemples sont manquants.

10.4.5 Sélection des verbatims à présenter au codificateur

Au delà des problématiques de traitement automatique des messages d'opinions, nous avons été confrontés au problème de l'optimisation des verbatims à présenter au codificateur.

Ce problème est important dans un contexte industriel lié à une volonté de réduire le coût humain des codifications (i.e., classification manuelle). Ce problème, déjà présent lors de traitements ponctuels, s'est trouvé renforcé lors du passage sur des traitements continus.

Cependant, aucune stratégie de sélection des verbatims à présenter au codificateur n'était jusque là employé. Le codificateur avait alors toute liberté de valider / codifier manuellement tout ou partie du traitement effectué au préalable par les algorithmes.

La stratégie mise en place consiste simplement alors à valider directement les verbatims déjà rencontrés au préalable. Ces verbatims ne sont alors plus accessibles aux codificateurs pour validation manuelle.

Cette stratégie peut paraître de prime abord assez simpliste et donc se révéler peu efficace. Afin de la valider à moindre coût de développement, nous avons choisi de ne retenir que 50 verbatims par idées comme messages de référence pour la validation. Ces 50 verbatims sont choisis selon les 3 critères qu'ils :

– ne comportent qu'une seule idée ;

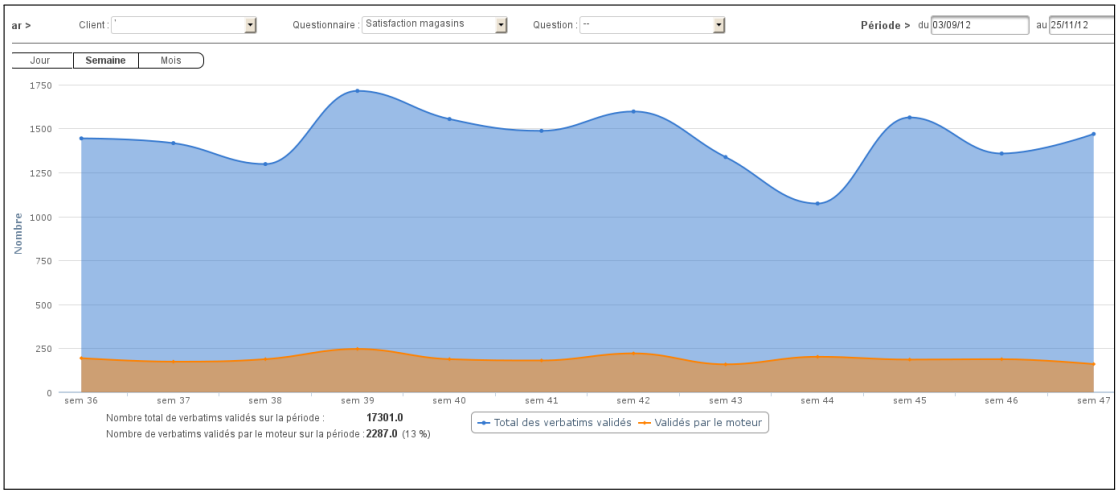


FIGURE 10.24 – Graphique de statistiques de validation sur un questionnaire de type NPS pour un réseau de magasin de vêtements par semaine entre le 03 septembre et le 25 novembre 2012. Les courbes présentent en abscisse les numéros de chaque semaine dans la période et en ordonnée le nombre de verbatims traités automatiquement (i.e., sans aucune revue manuelle) et le nombre de verbatims traités au total. Sur les 17301 verbatims traités sur cette période, 13% ont été directement validés par le système sans requérir une intervention humaine, soit 2287 verbatims.

- sont simples (i.e., ils ne comportent pas de caractères de ponctuation forte à part à la fin) ;
- sont répétés fréquemment.

Ces critères sont choisis pour sélectionner les meilleurs messages de référence afin d'obtenir un ratio validation / coût de traitement satisfaisant.

Des mesures de la performance de cette sélection ont ainsi été réalisées en production sur 3 questionnaires de 3 clients différents dont 2 enquêtes de réseaux de magasins de vêtements et une enquête après contact avec le service réclamation. Tous ces questionnaires sont de type NPS.

Les figures 10.24, 10.25, 10.26, 10.27, 10.28, 10.29 et 10.30 présentent les statistiques de validation sur ces 3 questionnaires par semaine entre le 03 septembre et le 25 novembre 2012.

Pour le questionnaire pour le 1er réseau de magasin de vêtements (figure 10.24), les statistiques par semaine sont stables : environ 1500 verbatims sont traités toutes les semaines et environ 200 (13%) en moyenne sont validés.

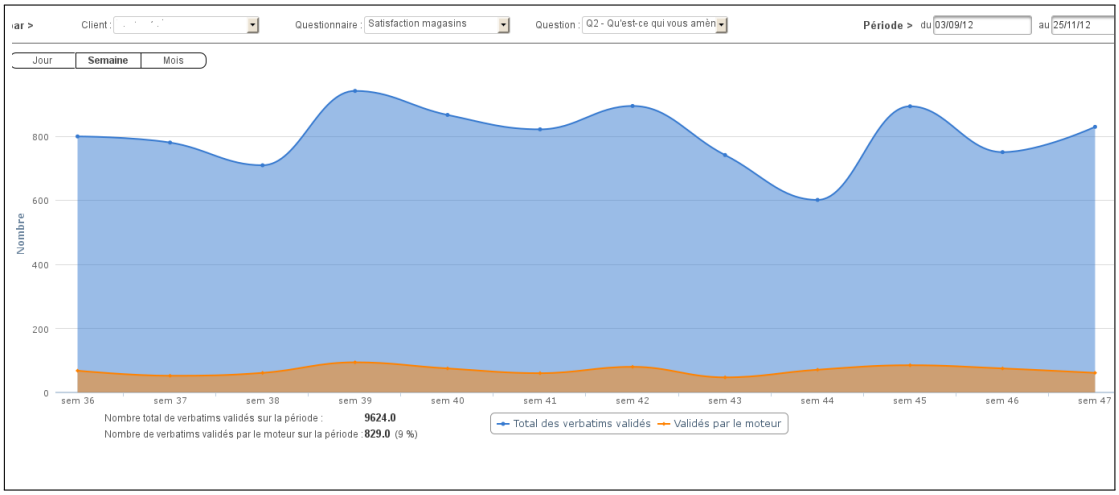


FIGURE 10.25 – Graphique de statistiques de validation sur la question 2 d'un questionnaire de type NPS pour un réseau de magasin de vêtements par semaine entre le 03 septembre et le 25 novembre 2012. Les courbes présentent en abscisse les numéros de chaque semaine dans la période et en ordonnée le nombre de verbatims traités automatiquement (i.e., sans aucune revue manuelle) et le nombre de verbatims traités au total. Sur les 9624 verbatims traités sur cette période, 9% ont été directement validés par le système sans requérir une intervention humaine, soit 829 verbatims.

Pour la question 2 du questionnaire pour le 1er réseau de magasin de vêtements (figure 10.25), les statistiques par semaine sont stables : environ 800 verbatims sont traités toutes les semaines et environ 70 (9%) en moyenne sont validés.

Pour la question 3 du questionnaire pour le 1er réseau de magasin de vêtements (figure 10.26), les statistiques par semaine sont stables : environ 650 verbatims sont traités toutes les semaines et environ 120 (19%) en moyenne sont validés.

Pour le questionnaire pour le 2e réseau de magasin de vêtements (figure 10.27), les statistiques par semaine sont stables : environ 1500 verbatims sont traités toutes les semaines et environ 210 (14%) en moyenne sont validés.

Pour la question 2 du questionnaire pour le 2e réseau de magasin de vêtements (figure 10.28), les statistiques par semaine sont stables : environ 900 verbatims sont traités toutes les semaines et environ 80 (9%) en moyenne sont validés.

Pour la question 3 du questionnaire pour le 2e réseau de magasin de vêtements (figure 10.29), les statistiques par semaine sont stables : environ 650 verbatims sont traités toutes les semaines et environ 143 (22%) en moyenne sont validés.

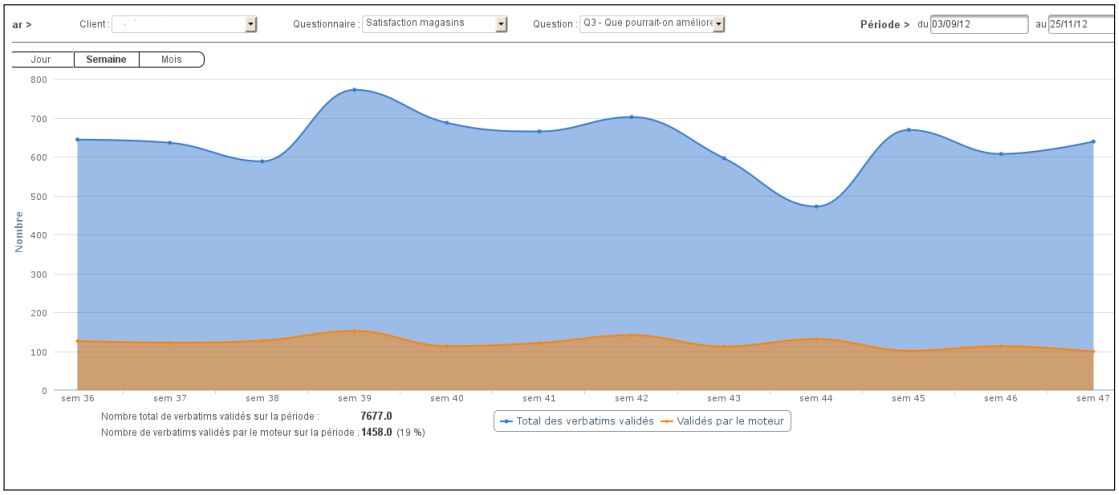


FIGURE 10.26 – Graphique de statistiques de validation sur la question 3 d'un questionnaire de type NPS pour un réseau de magasin de vêtements par semaine entre le 03 septembre et le 25 novembre 2012. Les courbes présentent en abscisse les numéros de chaque semaine dans la période et en ordonnée le nombre de verbatims traités automatiquement (i.e., sans aucune revue manuelle) et le nombre de verbatims traités au total. Sur les 7677 verbatims traités sur cette période, 19% ont été directement validés par le système sans requérir une intervention humaine, soit 1458 verbatims.

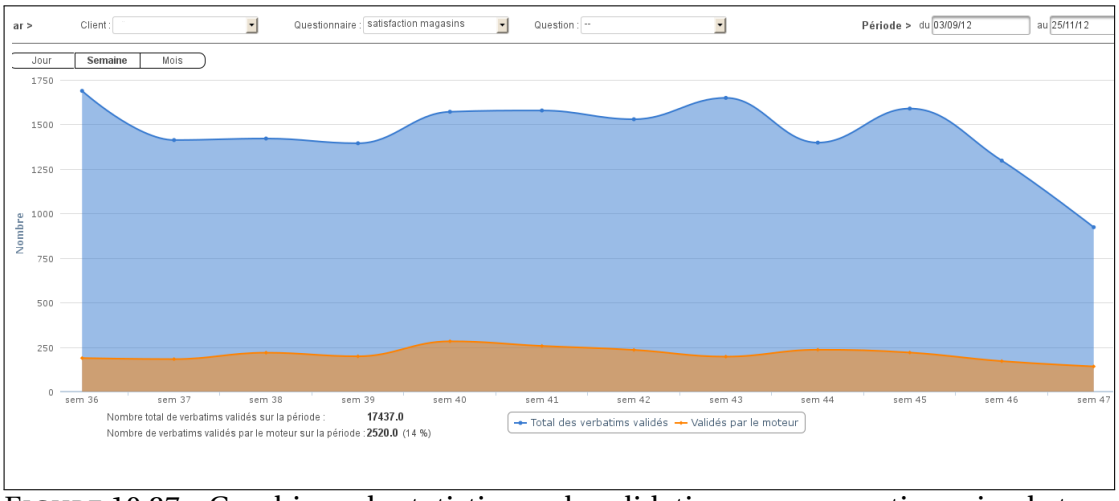


FIGURE 10.27 – Graphique de statistiques de validation sur un questionnaire de type NPS pour un deuxième réseau de magasin de vêtements par semaine entre le 03 septembre et le 25 novembre 2012. Les courbes présentent en abscisse les numéros de chaque semaine dans la période et en ordonnée le nombre de verbatims traités automatiquement (i.e., sans aucune revue manuelle) et le nombre de verbatims traités au total. Sur les 17437 verbatims traités sur cette période, 14% ont été directement validés par le système sans requérir une intervention humaine, soit 2520 verbatims.

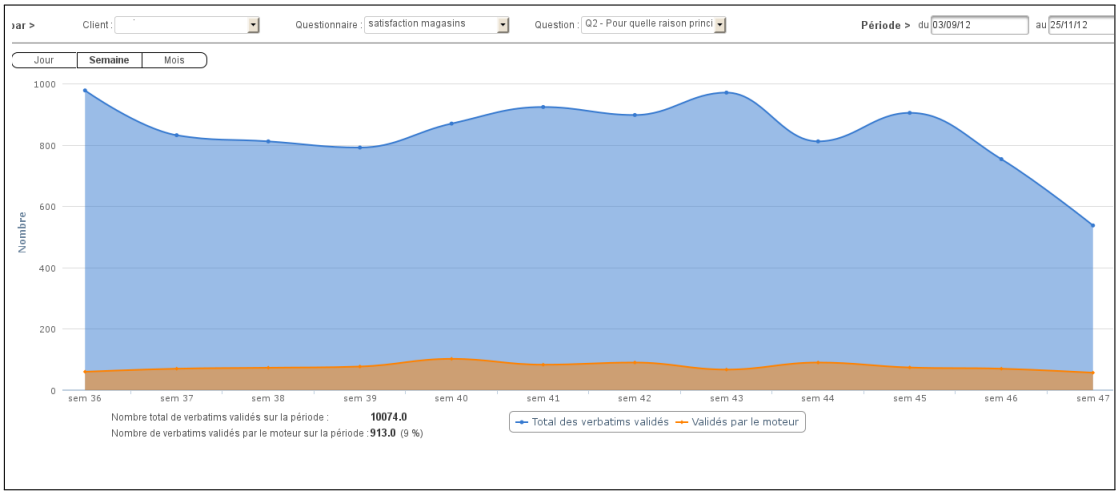


FIGURE 10.28 – Graphique de statistiques de validation sur la question 2 d'un questionnaire de type NPS pour un deuxième réseau de magasin de vêtements par semaine entre le 03 septembre et le 25 novembre 2012. Les courbes présentent en abscisse les numéros de chaque semaine dans la période et en ordonnée le nombre de verbatims traités automatiquement (i.e., sans aucune revue manuelle) et le nombre de verbatims traités au total. Sur les 10074 verbatims traités sur cette période, 9% ont été directement validés par le système sans requérir une intervention humaine, soit 913 verbatims.

Pour le questionnaire après contact avec le service réclamation (figure 10.30), contrairement aux réseaux de magasins, les statistiques par semaine sont assez instables : le nombre de verbatims traités par semaine varie entre 10 et 300 verbatims. Le taux de validation est toujours très bas. Un maximum de 10 verbatims seulement est validé toute les semaines.

D'une manière générale, ce principe de validation par verbatim déjà rencontré se révèle pertinent pour des traitements quotidiens et dont les propos ont une tendance répétitive. Cela n'est pas souvent le cas lorsqu'il s'agit de traitements de réclamations où les répondants relatent leurs propres problèmes. En revanche, c'est le cas lorsqu'il s'agit de traitements de réseaux où les répondants jugent les prestations sur des critères communs (prix du service, amabilité du personnel, ...).

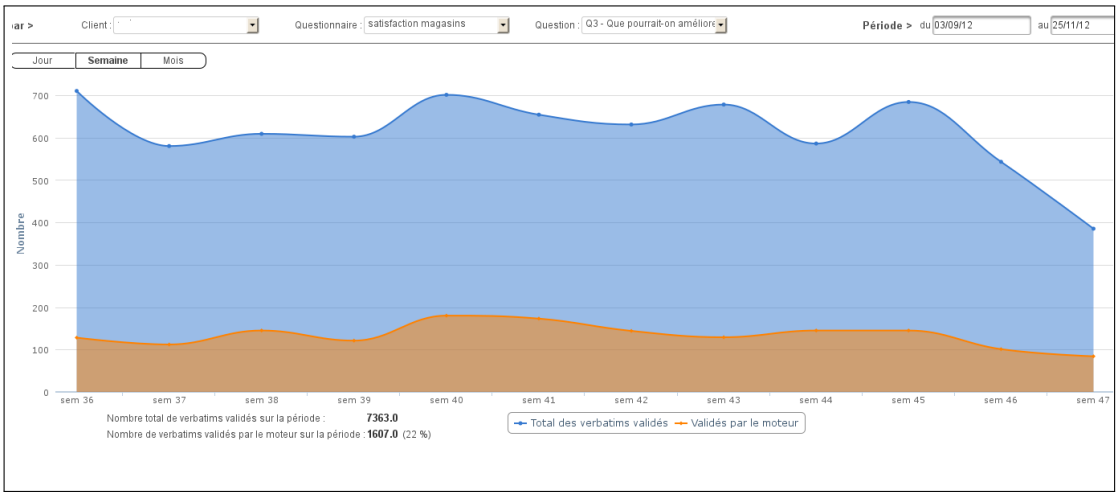


FIGURE 10.29 – Graphique de statistiques de validation sur la question 3 d'un questionnaire de type NPS pour un deuxième réseau de magasin de vêtements par semaine entre le 03 septembre et le 25 novembre 2012. Les courbes présentent en abscisse les numéros de chaque semaine dans la période et en ordonnée le nombre de verbatims traités automatiquement (i.e., sans aucune revue manuelle) et le nombre de verbatims traités au total. Sur les 7363 verbatims traités sur cette période, 22% ont été directement validés par le système sans requérir une intervention humaine, soit 1607 verbatims.

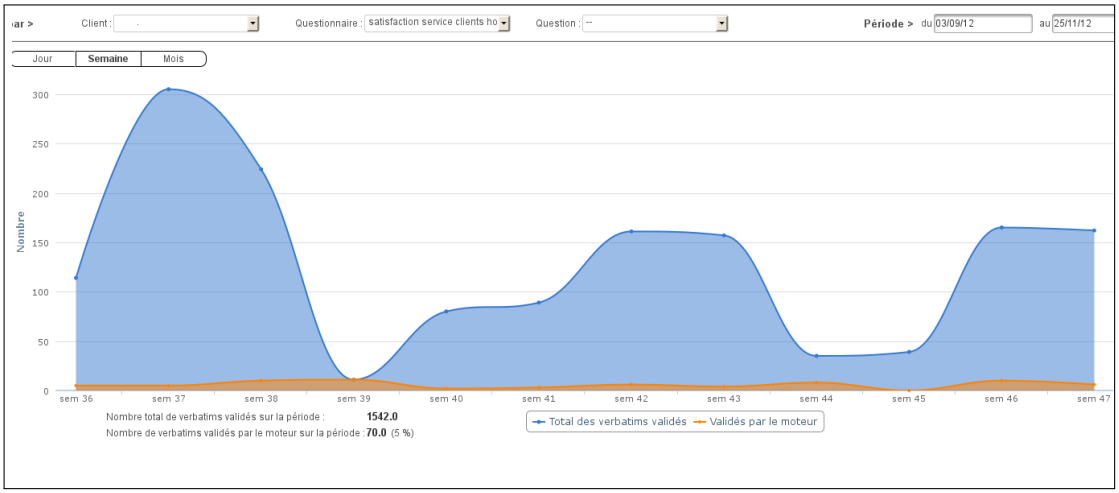


FIGURE 10.30 – Graphique de statistiques de validation sur un questionnaire de type NPS, après contact avec le service réclamation, par semaine entre le 03 septembre et le 25 novembre 2012. Les courbes présentent en abscisse les numéros de chaque semaine dans la période et en ordonnée le nombre de verbatims traités automatiquement (i.e., sans aucune revue manuelle) et le nombre de verbatims traités au total. Sur les 1542 verbatims traités sur cette période, 5% ont été directement validés par le système sans requérir une intervention humaine, soit 70 verbatims.

Chapitre 11

Recherche de produits dans une base de données à partir de requêtes en texte libre

La première section de ce chapitre a fait l'objet d'une publication à la conférence internationale *Global Wordnet Conference* (GWC), édition 2012, Matsue, Japon (TROUVILLIEZ 2012).

La constitution automatique de ressources spécifiques riches d'un point de vue linguistique reste un défi majeur dans les domaines du traitement des langues et des terminologies. La collecte d'informations linguistiques peut être manuelle si les ressources humaines et le temps de traitement ne sont pas prohibitifs. De ce fait, c'est une approche possible pour la constitution de ressources généralistes stables. Quand un traitement manuel n'est pas envisageable, il est nécessaire de réaliser ce traitement de manière automatique ou semi-automatique. Nous proposons d'utiliser un tel processus pour constituer automatiquement une ressource destinée à rendre possible la recherche de produits dans une base de données par des requêtes en texte libre. Nous utilisons ainsi une ressource linguistique généraliste pour étendre la connaissance du catalogue. Il s'agit de connecter le vocabulaire spécifique du catalogue dans la base de données et le vocabulaire de la ressource linguistique généraliste. Ce chapitre décrit les traitements et processus de liaison. Le système résultant

est évalué dans un contexte de recherche de produits utilisant des requêtes en texte libre. Cette recherche est basée sur la ressource linguistique construite par le processus décrit dans la suite.

11.1 Ressource pour la recherche de produits

11.1.1 Introduction

Dans le début des années 90, les chercheurs dans le domaine de la construction de ressources terminologiques travaillèrent principalement sur la constitution de ressources spécifiques à des domaines (RASTIER 1991). De nos jours, ces ressources sont tournées vers les tâches à accomplir. Dans (BOURIGAULT, AUSSENAC-GILLES et CHARLET 2004), des ressources spécifiques sont construites pour des applications dans le médical, le judiciaire ou l'industrie, basées sur des informations extraites de *corpora* spécialisés et complétées si besoin par des experts. Notre ambition est de réaliser l'extraction et la complétion de ces informations de manière automatique. Cela nécessite la liaison automatique de vocabulaires (lexiques) généraux et spécifiques à la tâche. La ressource spécifique résultante est bien sûr moins précise et fiable que celle construite par un expert mais elle est aussi moins coûteuse à construire et à améliorer.

Nous extrayons ainsi des informations à partir de catalogues de vente contenant différents produits avec de multiples attributs (prix, couleur, marque,...) puis nous les enrichissons par le biais de la connaissance contenue dans des ressources linguistiques généralistes. La ressource spécifique résultante nous permet ainsi d'effectuer des recherches naïves en langage naturel sur les produits.

La première section est dédiée à la mise en évidence de la nécessité d'enrichir linguistiquement les informations des catalogues de vente afin d'effectuer des recherches pertinentes en texte libre. Dans la seconde section, nous présentons notre stratégie de construction de ressources spécifiques. Nous comparons notre approche par enrichissement linguistique de catalogues de vente avec d'autres méthodes existantes et montrons les bénéfices obtenus. La dernière section est consacrée à l'éva-

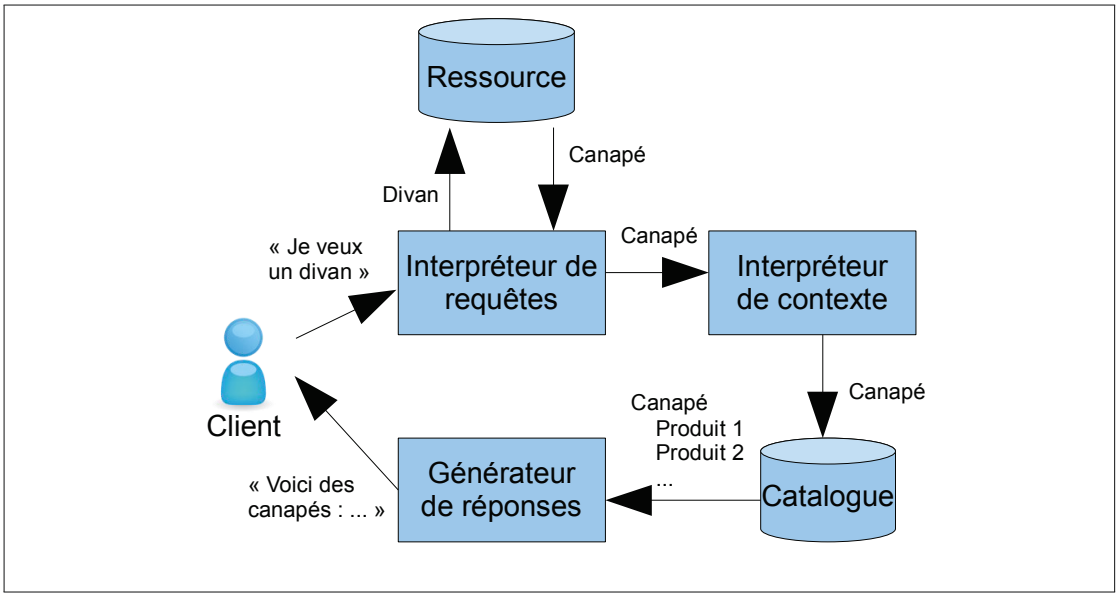


FIGURE 11.1 – Requêtes sur des catalogues de vente. Le client émet une requête qui est interprétée avec l'aide de la ressource. Après application du contexte, un filtre de produits est extrait de la base de produits et une réponse est générée pour envoi à l'internaute.

luation de la ressource spécifique et de sa capacité à fournir des réponses pertinentes aux requêtes.

L'étude fut réalisée principalement pour le français mais des résultats et exemples sont fournis pour l'anglais.

11.1.2 Nécessité de l'enrichissement linguistique : étude dans le cadre de catalogues de vente

Cadre général

Le travail décrit dans ce chapitre est une partie d'un projet ambitieux dont le but est la construction d'un agent conversationnel vendeur sur Internet. Ce chapitre s'intéresse à la problématique de construction de la ressource nécessaire pour la transcription d'une recherche de produit en langage naturel en une requête interprétable par un système de gestion de base de données. La figure 11.1 illustre le processus.

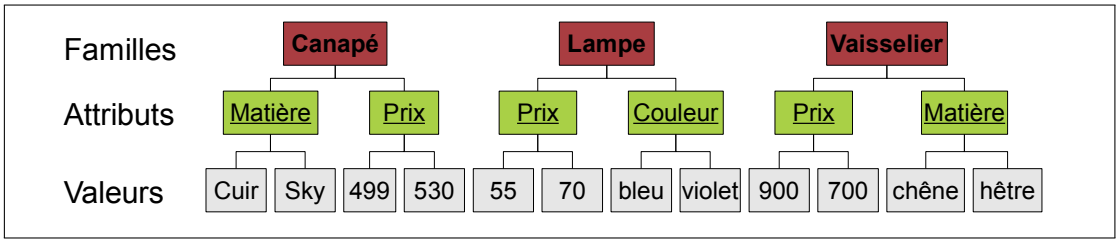


FIGURE 11.2 – Structure simplifiée du catalogue de vente. Un exemple est ici donné sur 3 familles contenant chacune 2 attributs avec 2 valeurs possibles pour chacun d'eux. Soient 3 familles, 6 attributs et 12 valeurs.

Yaourtiere
Tringle, barre a rideau
Vidéoprojecteur
Tiroir
Wok, tajine
Tondeuse homme
Vêtement de travail homme
Thermomètre bébé
Chaine hifi

TABLE 11.1 – Libellés de familles

Sources primaires d'informations : les catalogues de vente

La base de données contenant le catalogue est la seule ressource reliée à la tâche dont nous disposons. Elle a été construite par un processus semi-automatique et fournit une taxonomie basique pour la constitution du vocabulaire spécifique. Les produits sont organisés en familles telles que celles des canapés, des lampes ou des chemises (Table 11.1). Ces familles correspondent la plupart du temps à des rayonnages de magasin. Les produits sont caractérisés par des attributs (Table 11.2) propres à leur famille et possédant chacun une valeur particulière permettant de les différencier (Table 11.3). Ces valeurs sont issues de traitements semi-automatiques appliqués à des données brutes lors de la construction du catalogue. Les informations fournies peuvent être partielles. La figure 11.2 illustre la taxonomie.

Libellé de Famille	Libellé d'Attribut	Unité d'Attribut
Chaine hifi	Profondeur	mm
Chaine hifi	Hauteur	mm
Chaine hifi	Largeur	mm
Chaine hifi	Prix	€
Chaine hifi	Marque	
Chaine hifi	Wifi	
Chaine hifi	Enregistreur MP3	
Chaine hifi	HDMI	
Chaine hifi	Lecteur/Enregistreur DVD	
Chaine hifi	Description	
Chaine hifi	Référence	
Chaine hifi	Watts	

TABLE 11.2 – Attributs de la famille *Chaine hifi*. En colonne sont présentés les libellés de famille et d'attribut ainsi que l'unité d'attribut. Chaque ligne constitue un attribut de la famille *Chaine hifi*.

Libellé d'Attribut	***Chaine hifi* 1 values**	***Chaine hifi* 2 values**	***Chaine hifi* 3 values**
Marque	Philips	Philips	Muse
Prix	129.00	168.00	128.00
Référence	*Chaine Hifi DVD USB Philips MCD122*	*Chaine Hifi DVD USB Philips MCD712*	*Microchaine DVD MP3 Muse M-68DM*
Description	*Lecture via USB direct et une expérience sonore 2x20W*	*Intisifiez votre expérience sonore avec un son 2x50W et une lecture USB directe*	*Une qualité d'écoute de 2x20W dans un design épuré*

TABLE 11.3 – Valeurs d'attributs pour les produits de la famille *Chaine hifi*. En colonne sont présentés les libellés d'attributs ainsi que les valeurs pour 3 produits de la famille *Chaine hifi*. Chaque ligne constitue un attribut de la famille *Chaine hifi* avec les 3 valeurs de cet attribut pour les 3 produits.

Limites linguistiques des informations

Le catalogue a été construit pour la tâche de vente et ne sert donc à rien si l'on recherche autre chose que le libellé exact du produit ou une partie de celui-ci.

Certains problèmes ne peuvent être surmontés en s'appuyant uniquement sur le catalogue de par sa pauvreté linguistique. Par exemple, la requête *serviettes* ne renvoie aucune réponse bien que le catalogue contienne des *draps de bain*. Cette exemple illustre les problèmes liés à la synonymie. De plus, aucune généralisation ou spécialisation ne peut être faite. Ainsi, aucune réponse ne peut être apportée si l'on recherche une armoire dans un catalogue contenant des vaisseliers. Le même phénomène se produit si le catalogue contient des guitares et que l'on recherche des ukulélés. Ces problèmes doivent impérativement être corrigés car les clients n'ont aucun moyen de connaître les types exacts des produits vendus ou les dénominations utilisées dans le catalogue. De plus, contraindre ainsi les recherches possibles conduirait à une dégradation considérable de l'interaction avec l'agent.

Sources d'enrichissement linguistiques : les ressources généralistes

Nous arrivons ainsi à la conclusion que la connaissance contenue dans le catalogue doit être enrichie linguistiquement par le biais d'une mise en relation avec un lexique généraliste. Nous présentons à cette fin quelques ressources généralistes pouvant être utilisées dans la constitution de ce lexique.

Des ressources généralistes sont accessibles librement sur Internet. L'encyclopédie Wikipédia a ainsi été utilisée pour acquérir de la connaissance sur les erreurs les plus courantes en français dans (WISNIEWSKI, MAX et YVON 2010) et pour transcrire une ressource anglaise automatiquement en français dans (SAGOT et FIŠER 2008). Ce type de ressource, bien que contenant des informations linguistiques, reste cependant difficile à exploiter en terme d'extraction des dites informations de par leur richesse encyclopédique.

Des ressources construites semi-manuellement par des linguistes spécialement pour décrire les langues sont plus aisément exploitables dans ce contexte. La plus largement utilisée et la plus reconnue est le *Wordnet* de Princeton (MILLER 1995), créée pour la langue anglaise (voir section 10.1.2).

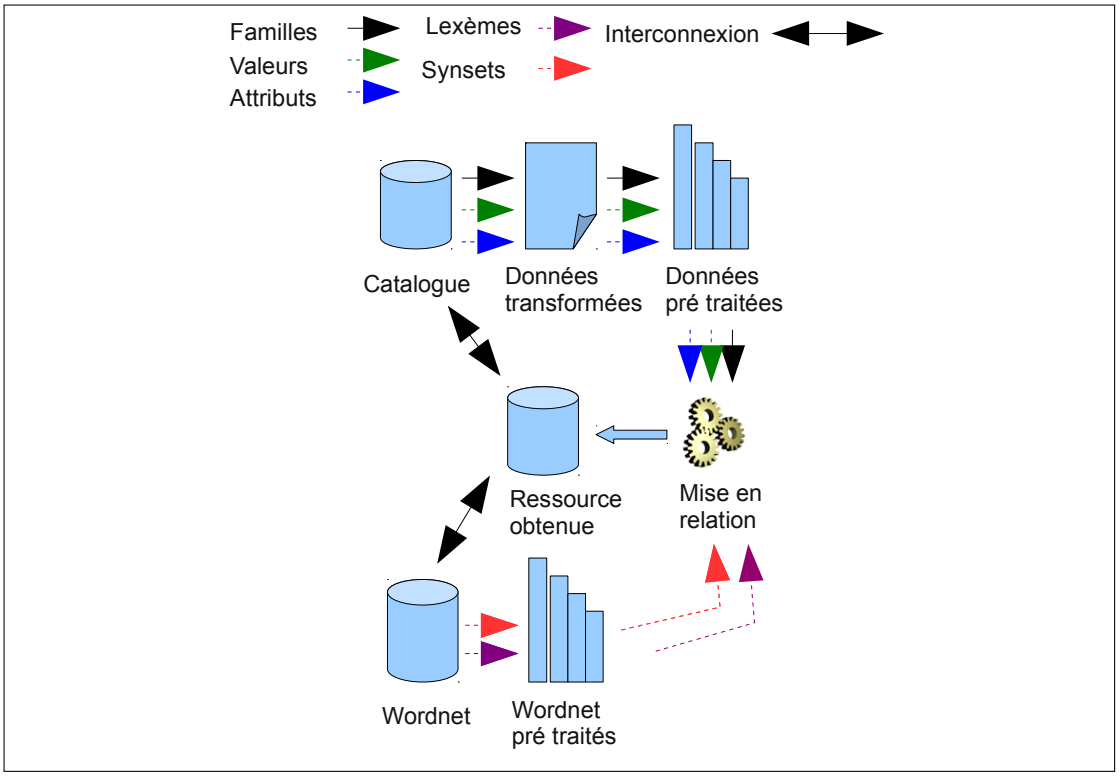


FIGURE 11.3 – Diagramme du processus de liaison des familles, attributs et valeurs d'un catalogue de produits avec les lexèmes des *synsets* de Wordnet. Les familles, attributs et valeurs sont extraites du catalogue pour être transformées et pré-traitées. En parallèle, les *synsets* et lexèmes sont extraits de Wordnet pour subir eux aussi un pré-traitement. Les données pré-traitées de part et d'autres sont mises en relation pour former une ressource spécialisée.

La dernière approche semble la meilleure pour nos travaux du fait de la présence explicite d'une structure sur les concepts de la langue qui peut être utilisée pour déterminer la proximité entre les différents termes mis en jeu dans la recherche. Nous procédons avec le WOLF (SAGOT et FIŠER 2008) et l'*EuroWordnet* FR (VOSSEN 1998) pour la langue française et avec l'*EuroWordnet* EN (VOSSEN 1998) pour l'anglais. Nous discutons de ce choix plus précisément à la section 11.1.3. Pour le moment, nous admettons simplement que nous constituons des lexiques généralistes à partir de ces ressources.

11.1.3 Intégration de connaissances généralistes

Des liens doivent être établis entre les informations du catalogue et les informations des ressources généralistes. Pour cela, une mise en relation automatique doit être réalisée entre les lexiques spécifiques et généralistes extraits de part et d'autre. La figure 11.3 illustre ce processus que nous décrivons en détail dans cette section.

Extraction / construction du lexique spécifique

Nous devons extraire les informations pertinentes des catalogues pour l'interprétation des recherches. Nous identifions trois types de champs :

1. libellés de familles et d'attributs (canapé, lampe,...) (matière, prix,...)
2. unités d'attributs (€, cm,...)
3. valeurs pour chaque attribut de chaque produit (cuir, chêne,...)

Les informations sélectionnées doivent être nettoyées afin d'être adaptées à la tâche. Ces informations constituent alors le lexique spécifique.

Adaptation spécifique de la taxonomie Une taxonomie généraliste devrait posséder une famille pour chaque type de vêtement (par exemple, « chemises », « pantalons ») et un attribut genre précisant à quel genre s'adresse le vêtement (par exemple, « homme », « femme », « enfant »). Mais, pour des raisons liées au marketing, les familles comportent directement l'attribut genre dans leur dénomination (par exemple, « chemise homme », « chemise femme ») afin de suggérer un rayonnage. Cela crée des relations non canoniques entre les lexiques spécifiques et généralistes.

Afin de contourner le problème, la présence inadaptée de mots tels que homme, femme, enfant dans les dénominations de familles doit être gérée. Un attribut genre est ainsi ajouté à ces familles et complété avec les valeurs appropriées pour chaque produit. La table 11.4 illustre ce filtrage.

De façon similaire, pour des raisons marketings, les dénominations de familles contiennent des listes de concepts séparés par des virgules. La famille T- shirt, veste est un exemple. Nous supposons qu'il est intéressant de transformer ces associations

Libellé de famille	Libellé nettoyé	Termes trouvés			Termes après racinisation		
Drap plat sheet	*Drap plat* sheet	*Drap plat* sheet			*Drap plat* sheet		
Manteau, veste femme woman coat, jacket	*Manteau, veste* coat, jacket	*Manteau* coat	*veste* jacket		*Manteau* coat	*veste* jacket	
Manteau, veste sport femme woman sport coat	*Manteau, veste sport* sport coat	*Manteau* sport coat	*veste sport*		*Manteau* sport coat	*veste sport*	
Tee-shirt, polo, chemise tee-shirt, polo, shirt	*Tee-shirt, polo, chemise* tee-shirt, polo, shirt	*Tee-shirt* Tee-shirt	*polo* polo	*chemise* shirt	*Tee-shirt* Tee-shirt	*polo* polo	*chemise* shirt
Television, Tv	Television, Tv	Television	Tv		Television	Tv	
Tringle, barre a rideau curtain rod	*Tringle, barre a rideau* curtain rod	*Tringle* curtain rod	*barre a rideau*		*Tringle* curtain rod	*barre a rideau*	
Taie d'oreiller, de traversin pillow, bolster case	*Taie d'oreiller, de traversin* pillow, bolster case	*Taie d'oreiller* pillow	*de traversin* bolster case		*Taie d'oreiller* pillow	*de traversin* bolster case	
Tapis carpet	*Tapis* carpet	*Tapis* carpet			*Tapis* carpet	*Tapi*	
Tapis de bain shower mat	*Tapis de bain* shower mat	*Tapis de bain* shower mat			*Tapis de bain* shower mat	*Tapi de bain*	
Chaussures sport sports shoes	*Chaussures sport* sports shoes	*Chaussures sport* sports shoes			*Chaussures sport* sports shoes	*Chaussure sport* sport shoe	

TABLE 11.4 – Pré-traitements et adaptations de données de catalogues en français et en anglais. En colonne sont présentés successivement les libellés de familles, les libellés nettoyés, les termes trouvés à partir de ces libellés et les termes trouvés après leur stemmatisation. Chaque groupe de ligne représente le traitement d'un même libellé de famille en français et en anglais.

de concepts du catalogue en liens de synonymie dans la ressource finale. L'association de concept peut être due à une réelle synonymie comme c'est le cas pour la famille Télévision, Tv. L'exploitation des différences entre la synonymie et les associations de concepts est étudiée plus précisément dans la section dédiée au processus de liaison lui-même (cf. section 11.1.3). Afin de construire ces nouveaux liens de synonymie, les différents concepts doivent être séparés les uns des autres. La table 11.4 illustre des exemples obtenus en anglais et en français. Cependant, dans certains exemples comme les familles « taie d'oreiller, de traversin », notre segmentation donne des libellés incomplets en raison du phénomène d'élision de termes répétés. Le processus de segmentation échoue alors pour le libellé ayant subi l'élision. La gestion de mots tels que *de* ou *à* dans le libellé pourrait être envisagée.

Correction orthographique Nous gérons des données issues de sources fiables en terme d'orthographe puisqu'il s'agit de données marketing et de ressources généralistes largement utilisées. Cependant, certains champs peuvent contenir une casse majuscule alors qu'il s'agit de noms communs (par exemple, Talkie Walkie) ou bien être sans accents (par exemple, Television). Une correction orthographique semble nécessaire. Néanmoins, nous avons recours à une insensibilisation de la procédure aux accents et à la casse en raison de la qualité des ressources mais surtout pour des contraintes fortes en termes de coût de traitement (temps CPU et mémoire vive utilisée). Bien sûr, cette solution limite notre capacité de correction par rapport à des solutions plus classiques comme celles présentées auparavant.

Lemmatisation / racinisation Des valeurs de produits du catalogue sont renseignées au pluriel alors que les informations linguistiques sont au singulier. Pour contourner ce problème, il est usuel de procéder à une lemmatisation des formes du même mot vers sa forme canonique. L'outil *TreeTagger* (SCHMID 1994) peut réaliser cela automatiquement. Néanmoins, nous utilisons à la place un processus simplifié de racinisation en raison de la qualité des ressources utilisées et surtout des contraintes déjà mentionnées. Tous les mots se terminant par 's' ou 'x' sont considérés comme au pluriel et des correspondances sont recherchées *à la fois* avec les *termes originaux* et ceux *sans ce symbole final*. Cette solution limite bien sûr notre capacité de lemma-

tisation. Ainsi, nous essayons de lier le terme chevaux avec chevau au lieu de cheval. Cependant, garder dans tous les cas le terme d'origine permet de garantir une absence de dégradation du processus global par l'ajout de ce traitement dans la limite où le terme proposé comme lemme ne correspond pas à un terme existant non corrélé au terme d'origine. La table 11.4 montre des exemples de ce traitement.

Dans la suite, les champs textuels ainsi obtenus dans le lexique spécifique sont dénotés simplement par termes.

Extraction / construction du lexique généraliste

La seconde partie de la tâche concerne les ressources généralistes. Nous devons extraire les informations les plus pertinentes pour la tâche. Un lexique généraliste est ainsi construit et ne contient que les parties intéressantes des ressources pour la tâche.

Lexèmes Des études préliminaires ont montré que seuls les lexèmes nominaux sont pertinents pour les données spécifiques que nous manipulons. Eux seuls sont gardés et les autres lexèmes sont supprimés. La pertinence de ce traitement est bien sûr liée à notre tâche. De plus, si le lexème nominal « machine à coudre » n'est pas présent dans la ressource, ce filtre empêche de trouver un lien via le lexème « coudre ». Les lexèmes non nominaux peuvent ainsi être utilisés pour contourner un manque de lexèmes nominaux. Les deux types de ressources étudiés à la section 11.1.2 sont pertinents car ils contiennent des lexèmes.

Types de liens linguistiques Comme nous le montrons à la section 11.1.2, nous avons besoin d'informations sur la synonymie et la généralisation de concepts. Les ressources *Wordnet* possèdent plusieurs types de liens entre les concepts des langues dont ceux dont nous avons besoin. Seuls les liens de synonymie et d'hyper/hyponymie sont gardés. Encore une fois, la pertinence de ce traitement est liée à notre tâche. *Wordnet* semble cette fois meilleur car il fournit explicitement ces liens contrairement aux ressources telle que Wikipedia. Notons également que d'autres liens tels que la meronymie (lien entre le tout et la partie) et l'holonymie (lien entre une partie et le tout) pourraient être intéressants dans notre traitement.

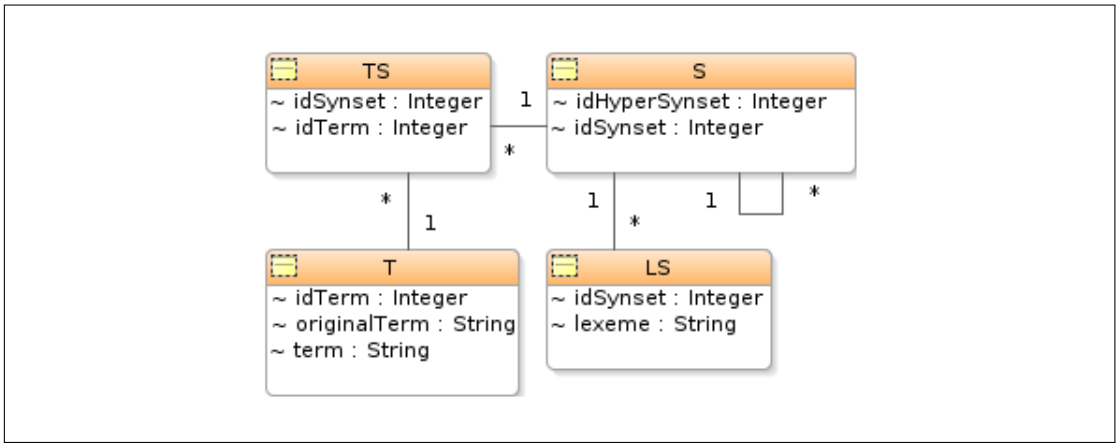


FIGURE 11.4 – Modèle relationnel

Intégration du lexique généraliste

Représentation relationnelle Soit T une relation représentative du lexique spécifique. Soient S et LS des relations représentatives des *synsets*, des items lexicaux (lexèmes), des liens entre les *synsets* et des liens entre les lexèmes et les *synsets* dans le lexique généraliste. La relation TS est remplie selon le processus décrit ci-dessous. La structure de données de ces relations est donnée à la figure 11.4. La table T est renseignée pour *idTerm* et *originalTerm* par les champs originaux correspondants dans le catalogue alors que le champ $term$ est complété par un des termes obtenus en sortie du traitement décrit à la section 11.1.3 pour le champ $originalTerm$ en

entrée. Autant de lignes que nécessaire sont insérées pour chaque entrée.

Input : T : relation des termes, S : relation des *synsets*, LS : relation de liaison des lexèmes et des *synsets*

Output : TS : relation de liaison des lexèmes et des *synsets*

1 **begin**

2 $TS \leftarrow \Pi_{idTerm,idSynset}(T \bowtie_{lexeme=term} LS)$;

3 $Words \leftarrow \{(idTerm, sterm) \mid \exists (idTerm', term) \in NLT, idTerm = idTerm' \wedge sterm \in split(term)\}$;

4 $WordsTLS \leftarrow \Pi_{idTerm,idSynset}(Words \bowtie_{lexeme=sterm} LS)$;

5 $SequenceTLS \leftarrow \{(idTerm, idSynset) \in NLT \times LS \mid \exists lexeme \in LS, isComplex(lexeme) \wedge (seq(lexeme) \subset seq(originalTerm))\}$;

6 $TLS \leftarrow WordsTLS \cup SequenceTLS$;

7 $TLSUID \leftarrow \rho_{idTerm,idHyperSynset,idSynset}\{(idTerm, idSynset, idSynset') \in TLS \times \mathbb{N} \mid \exists! idSynset' \in \mathbb{N}, idSynset' = uid(idSynset, idTerm)\}$;

8 $S \leftarrow S \cup \Pi_{idHyperSynset,idSynset}(TLSUID)$;

9 $TS \leftarrow TS \cup \Pi_{idSynset,idTerm}(TLSUID)$;

10 $TLS \leftarrow T \bowtie_{(originalTerm=lexeme) \vee (term=lexeme)} LS$;

11 $NewLTS \leftarrow \rho_{idTerm,newLabel}\left(\left(\Pi_{idTerm,originalTerm}(T) \cup \Pi_{idTerm,term}(T)\right) - \Pi_{idTerm,lexeme}(TLS)\right) \bowtie TS$;

12 $LS \leftarrow LS \cup \Pi_{idSynset,newLabel}(NewLTS)$;

13 **end**

Algorithme 2 : Algorithme de liaison. La relation TS est d'abord pré remplie avec les liens de synonymie stricte entre les deux lexiques (ligne 2). La relation $WordsTLS$ est ensuite remplie avec les liens d'hyperonymie entre les deux lexiques pour les termes simples (ligne 4) et la relation $SequenceTLS$ pour les termes composés (ligne 5). Les relations TS, TLS et LS sont ensuite mises à jour en fonction de ces deux dernières relations (lignes 9 à 12).

Avec :

$split(a)$ qui coupe une chaîne de caractères selon les espaces

$seq(a)$ qui retourne un ensemble ordonné de caractères selon leur ordre d'apparition dans la chaîne

$uid(a)$ qui retourne un identifiant unique

$isComplex(lexeme)$ qui retourne vrai si le lexème contient deux mots au moins

Intégration de la synonymie Les liens de synonymie sont intégrés. Des équivalences strictes sont recherchées entre les lexiques généralistes et spécifiques. La relation TS est remplie (algorithme 2, ligne 2).

Intégration de l'hyperonymie Les liens d'hyperonymie sont intégrés. Les hyponymies sont recherchées dans le lexique spécifique relativement au lexique généraliste. Par exemple, « chaussure de sport » contient strictement plus d'informations que « chaussure ». Nous supposons de manière générale qu'un label contenant strictement plus d'informations qu'un autre est un hyponyme du second. Deux cas peuvent se produire en fonction du lexème généraliste considéré :

- il est simple, i.e. non composé ;
- il est composé de deux mots au moins.

Sélection des données spécifiques pour l'intégration de l'hyperonymie L'intégration de l'hyperonymie est requise pour les entrées du lexique spécifique qui ne sont pas reliées par synonymie. La relation NLT est remplie avec ces entrées de la relation T.

Intégration de l'hyperonymie pour les lexèmes non composés La relation NLT contient des entrées pour lesquelles un des mots constituant les termes est connu en tant que donnée linguistique. Ces entrées et leurs correspondants linguistiques sont enregistrés dans la relation $WordsTLS$ (algorithme 2, ligne 4).

Intégration de l'hyperonymie pour les lexèmes composés La relation NLT contient des entrées pour lesquelles la séquence de caractères d'origine, correspondante au terme, inclut strictement une séquence de caractères correspondante à une donnée linguistique. Ces entrées et leur correspondant linguistique sont enregistrés dans la relation $SequenceTLS$ (algorithme 2, ligne 5). Cela est pertinent uniquement par son application exclusive aux lexèmes composés. Autrement, des associations erronées apparaissent par la possibilité d'identifier les lexèmes les plus courts dans de très nombreux mots non relatés. Le lexème « vie » pourrait ainsi être associé à de nombreux mots non pertinents dont « éVIEr » sans cette limitation.

Complétion des relations linguistiques Les relations linguistiques sont remplies en utilisant les deux sous-ensembles identifiés dans les précédentes étapes d'intégration de l'hyperonymie (algorithme 2, ligne 9).

Enrichissement des lexèmes Les informations connectées par synonymie et hyponymie sont utilisées pour enrichir les lexèmes. La relation T contient les termes qui ne correspondent pas à un label présent dans les ressources linguistiques. Néanmoins, un lien a été identifié entre la connaissance spécifique et les ressources linguistiques (relation TS). La relation LS est remplie avec ces informations (algorithme 2, ligne 12).

11.1.4 Temps et mode de traitement

L'environnement d'exécution requiert d'avoir un temps de traitement le plus restreint possible (idéalement inférieur à 10 minutes).

Nous postulons qu'une implémentation complète du processus sur les familles et partielle sur les attributs, unités et valeurs peut garantir ce temps de traitement tout en offrant un résultat pertinent. Cela inclut uniquement l'intégration de la synonymie pour les attributs, les unités et les valeurs.

Deux environnements sont considérés : un en temps limité dénoté par mode restreint et un autre non limité dénoté par mode normal. Les évaluations présentées ci-dessous comparent les résultats obtenus dans ces environnements.

11.1.5 Outil d'exploration de ressources linguistiques « *Wordnet* »

Une première démarche en terme d'évaluation a été la mise au point d'un outil permettant la visualisation graphique de notre ressource. Les figures 11.5, 11.6, 11.7 et 11.8 montrent les visualisations des lexèmes « pantalon » et « siège » dans la ressource linguistique *EuroWordnet* français et dans la ressource obtenue après application de notre procédure. La ressource obtenue contient plus de lexèmes dans le *synset* en rapport avec les pantalons que la ressource de départ. Les lexèmes « jean » et « short sport », affichés en vert ont été ajoutés à ce *synset* grâce à l'association du

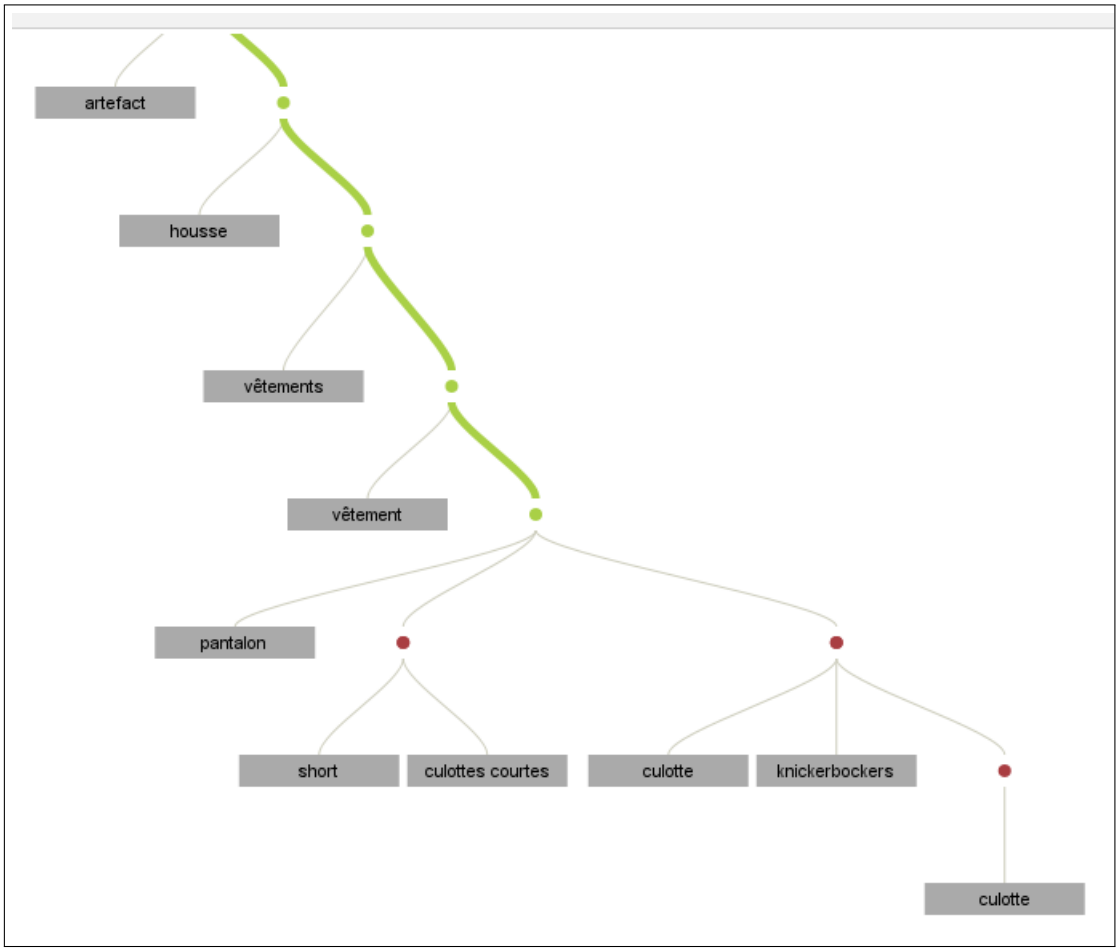


FIGURE 11.5 – Visualisation du lexème « pantalon » dans l'*EuroWordnet* français

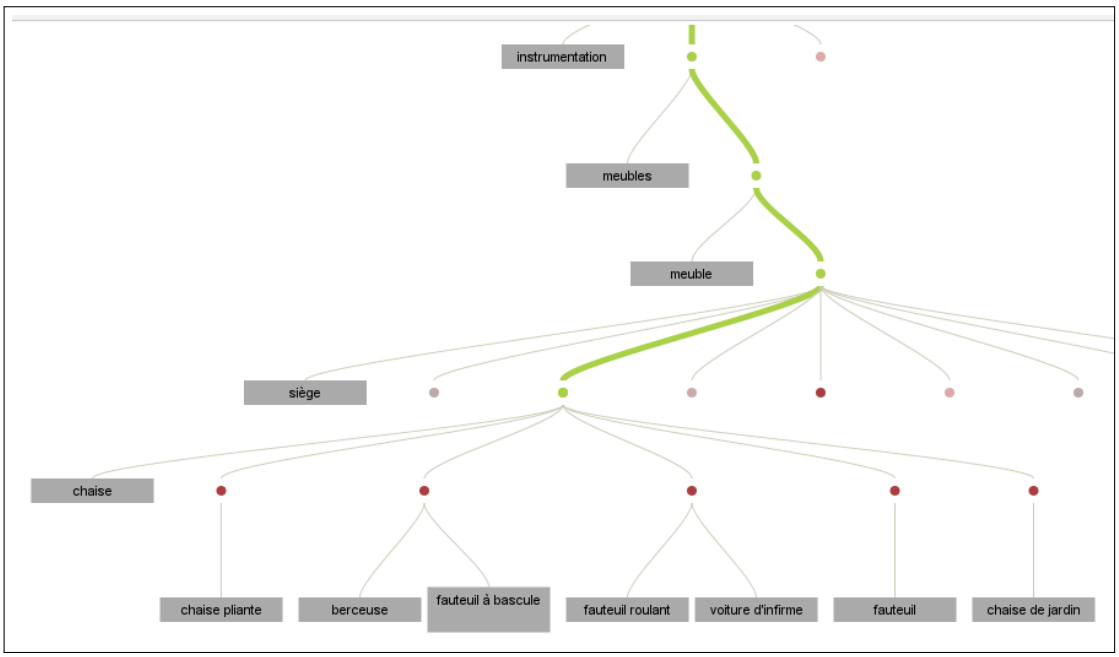


FIGURE 11.6 – Visualisation du lexème « siège » dans l'*EuroWordnet* français

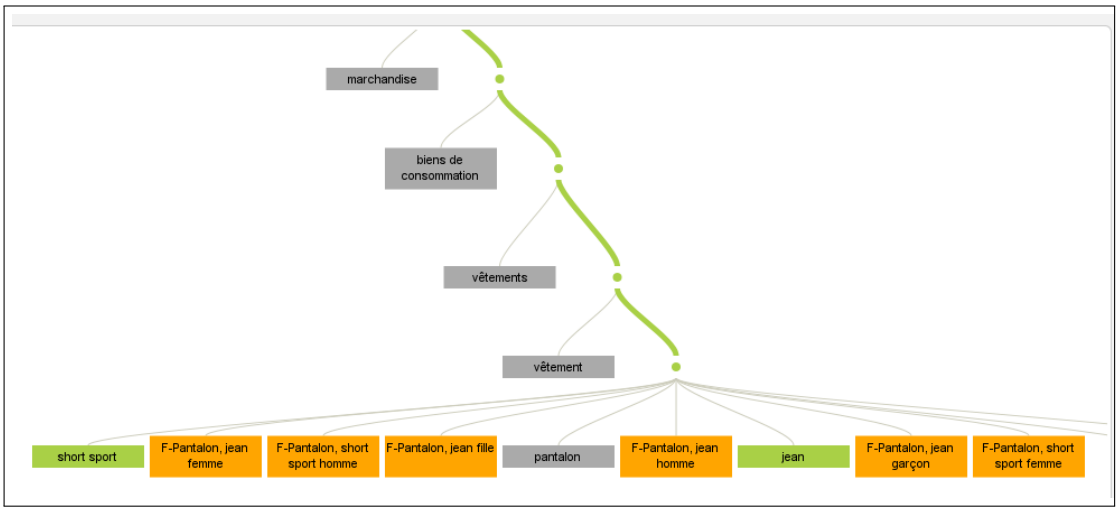


FIGURE 11.7 – Visualisation du lexème « pantalon » dans la ressource obtenue à partir de l'*EuroWordnet* français

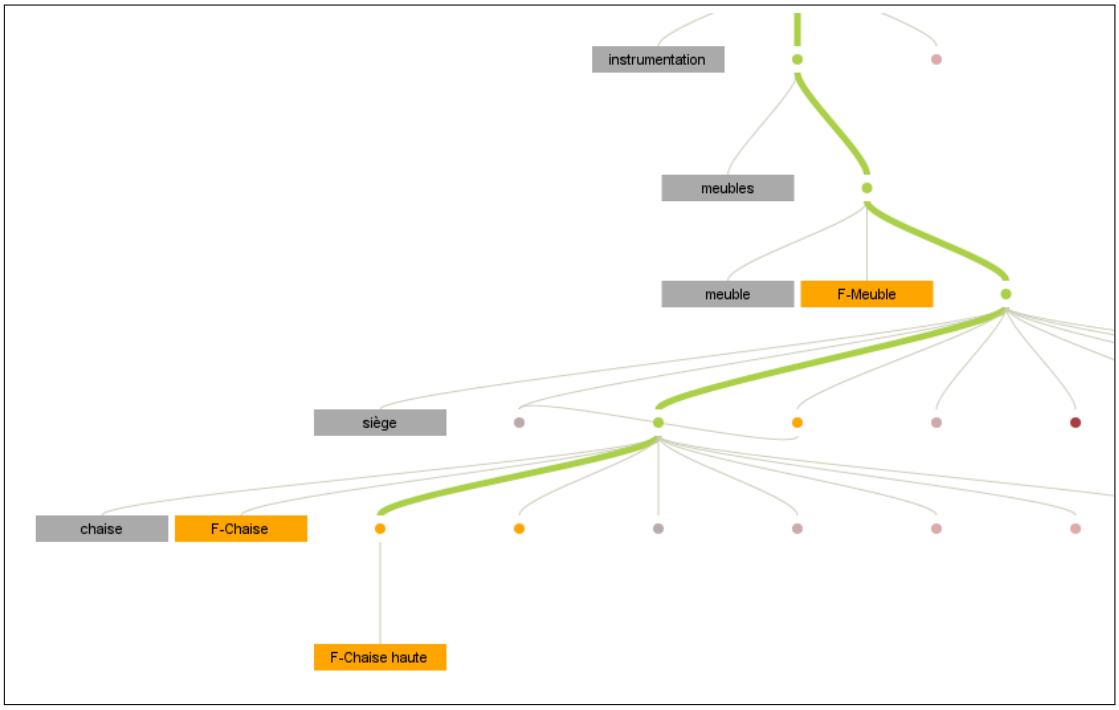


FIGURE 11.8 – Visualisation du lexème « siège » dans la ressource obtenue à partir de l'*EuroWordnet* français

synset contenant le lexème « pantalon » et des familles « pantalon, jean » et « pantalon, short sport » du catalogue, affichées en orange. De même, sur l'exemple du lexème « siège », on peut observer que la famille « meuble » a été trouvée en tant qu'hypéronyme (généralisation) tandis que celle des « chaises » a été trouvée en tant qu'hyponyme (spécialisation).

11.1.6 Évaluation de la ressource obtenue

Une évaluation statistique du processus est réalisée sur quatre catalogues en français portant sur différents domaines : un catalogue généraliste avec des vêtements, des accessoires, des ordinateurs, ... et trois catalogues spécialisés dans le vinicole, le high-tech et les voitures. Elle porte aussi sur un catalogue anglais spécialisé dans la vente d'articles de sport. Comme cette base ne contient ni attributs, ni unités en anglais, les résultats sont seulement calculés sur les familles et les valeurs. Les résultats sont donnés aux tables 11.5 et 11.6 et en mode normal par défaut. Les résul-

		catalogue généraliste			**catalogue vinicole**			**catalogue high-tech**			**catalogue voitures**		
		WOLF (W)	Euro-Wordnet (EWN)	W+EWN	W	EWN	W+EWN	W	EWN	W+EWN	W	EWN	W+EWN
Familles	#	273			7			92			6		
	# lié	198	226	237	4	4	4	66	68	75	3	3	4
	% lié	72.5%	82.8%	**86.9%**	**57.1%**	**57.1%**	**57.1%**	71.7%	73.9%	**81.5%**	50%	50%	**66.7%**
Attributs	#	12			16			43			56		
	# lié (partiel)	9 (7)	10 (8)	10 (8)	9 (7)	7 (5)	9 (7)	31 (20)	31 (20)	31 (20)	42 (12)	43 (13)	44 (13)
	% lié (partiel)	75% (58.3%)	**83.3%** (**66.7%**)	**83.3%** (**66.7%**)	**56.3%** (**43.8%**)	43.8% (31.3%)	**56.3%** (**43.8%**)	**72.1%** (**46.5%**)	**72.1%** (**46.5%**)	**72.1%** (**46.5%**)	75% (21.4%)	76.8% (**23.2%**)	**78.6%** (**23.2%**)
Valeurs	**# (partiel)**	21120 (3792)			8728 (360)			3612 (3496)			26337 (1420)		
	# lié (partiel)	13469 (232)	13743 (237)	16296 (331)	2686 (17)	4018 (13)	4462 (22)	565 (214)	414 (129)	674 (259)	2976 (127)	2889 (121)	3662 (180)
	% lié (partiel)	63.8% (6.1%)	65.1% (6.3%)	**77.2%** (**8.7%**)	30.8% (4.7%)	46% (3.6%)	**51.1%** (**6.1%**)	15.6% (6.1%)	11.5% (3.7%)	**18.7%** (**7.4%**)	11.3% (8.9%)	11% (8.5%)	**13.9%** (**12.7%**)
Unités	#	63			3			46			21		
	# lié (partiel)	27 (5)	21 (2)	30 (6)	3 (1)	2 (0)	3 (1)	22 (10)	17 (5)	23 (11)	10 (8)	2 (1)	11 (9)
	% lié (partiel)	42.9% (8.0%)	33.3% (3.2%)	**47.6%** (**9.5%**)	**100%** (**33.3%**)	66.7% (0%)	**100%** (**33.3%**)	47.8% (21.7%)	37% (10.9%)	**50%** (**23.9%**)	47.6% (38.1%)	9.5% (4.8%)	**52.4%** (**42.9%**)
Durée	**en mn (partiel)**	-	-	1046.87 (7.61)	-	-	627.95 (1.15)	-	-	929.05 (5.78)	-	-	333.93 (2)

TABLE 11.5 – Résultats statistiques de notre méthode de liaison sur des catalogues en français. Les résultats pour les 4 catalogues testés sont présentés en colonnes. Pour chacun d'eux, les résultats obtenus sur les liaisons des familles, des attributs, des valeurs et des unités sont présentés par groupes de lignes. Les résultats sont à chaque fois présentés pour le WOLF, l'*EuroWordnet* et la ressource constituée de l'union des deux. Les résultats partiels correspondent à ceux obtenus en « mode restreint » en opposition à ceux obtenus en « mode normal ». Les pourcentages de liaison des éléments de chaque type sont à chaque fois donnés par rapport au nombre total d'éléments du type considéré dans le catalogue.

		catalogue de sport
		EWN
Familles	**#**	52
	# lié	43
	% lié	**82.7%**
Valeurs	**# (partiel)**	5475 (645)
	# lié (partiel)	3767 (111)
	% lié (partiel)	**68.8%** (**17.2%**)
Durée	**en mn (partiel)**	333.28 (1.36)

TABLE 11.6 – Résultats statistiques de notre méthode de liaison sur un catalogue en anglais. Les résultats pour le catalogue testé sont présentés en colonnes. Les résultats obtenus sur les liaisons des familles et des valeurs sont présentés par groupes de lignes. Les résultats sont présentés pour l'*EuroWordnet* seul. Les résultats partiels correspondent à ceux obtenus en « mode restreint » en opposition à ceux obtenus en « mode normal ». Les pourcentages de liaison des éléments de chaque type sont à chaque fois donnés par rapport au nombre total d'éléments du type considéré dans le catalogue.

tats en mode restreint, si différents, sont donnés entre parenthèses. Cette stratégie restreinte ne peut trouver de liens entre les ressources linguistiques et les labels et valeurs les plus complexes. Le nombre d'attributs et valeurs considérés pour l'évaluation est alors limité.

Les meilleurs résultats statistiques sur les catalogues français (en fonte grasse dans le tableau 11.5) indiquent que la ressource construite à partir du WOLF et de l'*EuroWordnet* est au moins aussi performante voire, dans la majorité des cas, plus performante que les ressources obtenues à partir de chacun des deux. La liaison du catalogue généraliste est légèrement meilleure que celle des trois catalogues spécialisés, en particulier pour les valeurs. Un autre point intéressant concerne la plus faible qualité des liaisons des caractéristiques, valeurs et unités dans le mode restreint, comparé au mode normal. Mais comme attendu, le processus est en moyenne 250 fois plus rapide en mode restreint qu'en mode normal. Les résultats relevés pour l'anglais sont assez similaires aux meilleurs résultats sur les cas français avec un pourcentage de liaison sur les familles de l'ordre de 80% et un pourcentage de liaison sur les valeurs de l'ordre de 70% en mode normal en dépit de la spécificité du catalogue sur les équipements sportifs. En mode restreint, on observe même une légère augmentation sur les valeurs avec un taux de succès de près de 20% contre 10% pour le meilleur cas français.

11.2 Recherche de produits

11.2.1 Description du processus

La recherche des produits elle-même est basée sur les 3 phases suivantes :

1. la reconnaissance des entités « probables » du catalogue ;
2. la liaison syntaxique des différentes entités ;
3. l'inférence des entités implicites dans la requête, la suppression éventuelle des entités détectées incohérentes et l'application de distributivité.

Nous présentons plus explicitement ces 3 phases dans les sous-sections suivantes.

Exemple 11.1

Pour illustrer nos propos dans cette partie, nous utilisons le texte « Je cherche un canapé en cuir à moins de 1500 euros et une lampe en cuir » comme texte saisi par l'internaute. Ce texte comporte volontairement une incohérence sur la lampe recherchée pour illustrer la gestion de ce cas.

Reconnaissance d'entités

Nous utilisons la ressource construite préalablement pour reconnaître des entités du catalogue :

- $FAMILLE$: trouve les chaînes correspondantes à un synonyme d'une famille définie dans le catalogue, à un hypéronyme ou hyponyme sur 3 niveaux d'une famille définie dans le catalogue ;
- $ATTRIBUT$: trouve les chaînes correspondantes à un synonyme d'un attribut défini dans le catalogue, à un hypéronyme ou hyponyme sur 3 niveaux d'un attribut défini dans le catalogue ;
- $VALEUR$: trouve les chaînes correspondantes à une donnée numérique, à un synonyme d'une valeur définie dans le catalogue, à un hypéronyme ou hyponyme sur 3 niveaux d'une valeur définie dans le catalogue ;
- $UNITE$: trouve les chaînes correspondantes à un synonyme d'une unité définie dans le catalogue.

Nous utilisons de plus l'analyseur syntaxique par dépendances *MaltParser* pour rechercher les entités « complexes » du catalogue.

Exemple 11.2

Sur notre texte exemple, cette reconnaissance correspond à l'étiquetage suivant :
Je cherche un $\boldsymbol{canapé}_{FAMILLE}$ *en* $\boldsymbol{cuir}_{VALEUR}$ *à moins de* $\mathbf{1500}_{VALEUR}$ $\boldsymbol{euros}_{UNITE}$ *et une* $\boldsymbol{lampe}_{FAMILLE}$ *en* $\boldsymbol{cuir}_{VALEUR}$.

Nous notons qu'il est possible que certaines entités du catalogue ne soient pas détectées alors qu'elles sont en relation de dépendance avec une autre entité détectée. Cela correspond à la notion d'entité implicite. Dans notre exemple, nous détectons des valeurs alors que nous ne détectons pas d'attributs : les attributs sont implicites.

En plus des entités du catalogue, il est nécessaire dans cette étape de reconnaître différents opérateurs :

- $CONJONCTION$ (par exemple, et) ;
- $ALTERNATIVE$ (par exemple, ou) ;
- $NEGATION$ (par exemple, ne...pas) ;
- $APPROXIMATION$ (par exemple, environ) ;
- $DELIMITEUR$ (par exemple, entre) ;
- $DELIMITEUR_{SUP}$ (par exemple, à moins de) ;
- $DELIMITEUR_{INF}$ (par exemple, à plus de).

Exemple 11.3

Sur notre texte exemple, l'étiquetage devient alors le suivant :
Je cherche un $canapé_{FAMILLE}$ *en* $cuir_{VALEUR}$ {**à *moins de***}$_{\boldsymbol{DELIMITEUR_{SUP}}}$ 1500_{VALEUR} $euros_{UNITE}$ $\boldsymbol{et_{CONJONCTION}}$ *une* $lampe_{FAMILLE}$ *en* $cuir_{VALEUR}$.

Liaison syntaxique des entités

Après avoir reconnu des entités, nous devons les lier entre elles pour former les requêtes « complexes » mettant en jeu des entités de différentes natures. Cette liaison est basée sur des patrons syntaxiques alliant les entités du catalogue aux opérateurs.

Pour notre exemple, nous avons recours aux trois patrons syntaxiques suivants :
Patron 1 : $CONJONCTION$, $FAMILLE$, $VALEUR$
Patron 2 : $FAMILLE$, $VALEUR$
Patron 3 : $DELIMITEUR_{SUP}$, $VALEUR$, $UNITE$

Exemple 11.4

Sur notre texte exemple, nous obtenons alors la reconnaissance de trois patrons :
$canapé_{FAMILLE}$, $cuir_{VALEUR}$
{à $moins\ de$}$_{DELIMITEUR_{SUP}}$, 1500_{VALEUR}, $euros_{UNITE}$
$et_{CONJONCTION}$, $lampe_{FAMILLE}$, $cuir_{VALEUR}$

Inférence des entités implicites, suppression des incohérences et distribution

Nous inférons ensuite les entités implicites du catalogue.

Exemple 11.5

Dans notre exemple, il s'agit d'inférer les attributs manquants. Nous obtenons alors les patrons suivants :
$canapé_{FAMILLE}$, $\boldsymbol{matière}_{\boldsymbol{ATTRIBUT}}$, $cuir_{VALEUR}$
$\boldsymbol{prix}_{\boldsymbol{ATTRIBUT}}$, {à $moins\ de$}$_{DELIMITEUR_{SUP}}$, 1500_{VALEUR}, $euros_{UNITE}$
$et_{CONJONCTION}$, $lampe_{FAMILLE}$, $\emptyset$, ~~$cuir_{VALEUR}$~~

Nous notons que l'inférence peut échouer à déterminer l'entité implicite comme c'est le cas ici pour la matière cuir de la lampe. Dans ce cas, le patron est réduit à sa sous-partie maximale cohérente et complète, ce qui revient dans notre exemple à ne conserver que la famille.

Exemple 11.6

Après suppression de l'incohérence, nous obtenons les patrons suivants :
$canapé_{FAMILLE}$, $matière_{ATTRIBUT}$, $cuir_{VALEUR}$
$prix_{ATTRIBUT}$, {à $moins\ de$}$_{DELIMITEUR_{SUP}}$, 1500_{VALEUR}, $euros_{UNITE}$
$et_{CONJONCTION}$, $lampe_{FAMILLE}$

Nous inférons enfin les entités « opérateurs » implicites et effectuons les distributions nécessaires pour compléter les patrons.

Exemple 11.7

Dans notre exemple, nous inférons la conjonction entre les deux caractéristiques du canapé :
$canapé_{FAMILLE}$, $matière_{ATTRIBUT}$, $cuir_{VALEUR}$
$\boldsymbol{et}_{\boldsymbol{CONJONCTION}}$, $\boldsymbol{canapé}_{\boldsymbol{FAMILLE}}$, $prix_{ATTRIBUT}$, {à $moins\ de$}$_{DELIMITEUR_{SUP}}$, 1500_{VALEUR}, $euros_{UNITE}$
$et_{CONJONCTION}$, $lampe_{FAMILLE}$

Corpus	catalogue généraliste	catalogue vinicole	catalogue high-tech	catalogue voitures
# messages	69	20	18	20
F-mesure sur les familles (partiel)	52% (50%)	42% (48%)	58% (62%)	32%
F-mesure sur les attributs (partiel)	50% (49%)	45% (49%)	65% (70%)	30%
% réponses attendues (partiel)	36% (38%)	20% (15%)	61% (67%)	15%
Temps pour tous les messages en mn (partiel)	2.21 (1.6)	1.03 (2.16)	1.33 (1.35)	0.15
Temps moyen pour un message en s (partiel)	1.9 (1.4)	3.1 (6.5)	4.4 (4.5)	0.5

TABLE 11.7 – Résultats en recherche de produits sur des catalogues en français. Les résultats pour les 4 catalogues testés sont présentés en colonnes. Pour chacun d'eux, les résultats obtenus sur la recherche de familles et d'attributs sont donnés ainsi que le pourcentage de réponses jugées exactes. Les résultats partiels correspondent à ceux obtenus en « mode restreint » en opposition à ceux obtenus en « mode normal »

Corpus	catalogue de sport
# messages	20
F-mesure sur les familles (partiel)	72% (78%)
F-mesure sur les attributs (partiel)	57% (62%)
% réponses attendues (partiel)	70% (65%)
Temps pour tous les messages en mn (partiel)	0.31 (0.35)
Temps moyen pour un message en s (partiel)	0.9 (1.1)

TABLE 11.8 – Résultats en recherche de produits sur un catalogue en anglais. Les résultats pour le catalogue testé sont présentés en colonnes. Les résultats obtenus sur la recherche de familles et d'attributs sont donnés ainsi que le pourcentage de réponses jugées exactes. Les résultats partiels correspondent à ceux obtenus en « mode restreint » en opposition à ceux obtenus en « mode normal »

11.2.2 Évaluation du processus de recherche de produits

Nous avons formé cinq *corpora* de textes constituants des requêtes de clients sur les cinq même catalogues que précédemment. Quatre sont donc en français et un en anglais. Chaque *corpus* est libellé manuellement en fonction des résultats attendus pour chaque requête. Nous évaluons les résultats retournés par le moteur de recherche en mode normal et restreint en accord avec trois critères : une F-mesure sur les familles, une F-mesure sur les attributs et le pourcentage des résultats retournés tels qu'attendus. Les tables 11.7 et 11.8 présentent ces résultats. Le catalogue anglais possède un score de F-mesure sur les attributs bien qu'il n'en contient pas. Ce score est calculé en utilisant les valeurs attendues dans la recherche. L'évaluation est faite avec la ressource construite à partir du WOLF et de l'*EuroWordnet* pour le français et à partir d'*EuroWordnet* pour l'anglais.

Les résultats montrent que notre système comprend correctement environ 30% des requêtes pour le français. Ceux-ci montrent aussi que le mode restreint (résultats entre parenthèses) obtient chaque fois un score similaire à celui obtenu en mode normal. La différence maximale obtenue est de 7%. La différence entre les deux modes dans l'évaluation précédente sur les attributs et les valeurs semble avoir un impact limité sur les réponses du système. Les performances de notre système apparaissent meilleures en anglais, avec 70% de réponses correctes sur le catalogue testé. Ce résultat est supérieur de 10% au meilleur score sur les catalogues en français.

11.3 Conclusion

Nous avons montré comment connecter les connaissances spécifiques et les ressources généralistes comme *Wordnet*. Les relations sont établies entre les deux en intégrant le lexique généraliste dans le spécifique. L'ajout de ressources linguistiques dans le processus tend à améliorer les résultats statistiques obtenus par le processus de liaison. Afin de limiter le temps de calcul, il est possible de restreindre le processus sans restreindre la qualité de la ressource obtenue au final. Ces résultats confirment également l'adaptabilité de notre procédure à d'autres langues que le français et confirment également que la qualité de la ressource obtenue est dépendante de la

qualité des ressources employées.

Chapitre 12

Gestion des couleurs dans les bases de données

Nombreuses sont les bases de données qui contiennent des informations de type couleur. Celles-ci sont usuellement représentées de façon textuelle, ce qui est problématique en terme de requêtes naïves car une mise en correspondance exacte des valeurs réduit typiquement de façon trop drastique l'ensemble des réponses attendues. Par exemple, rechercher du rouge dans une base ne contenant que du vermillon conduit à une réponse vide très peu pertinente. Si cette mise en correspondance peut être partiellement gérée par une analyse de synonymie et d'hypéronymie entre les expressions de couleurs, la gestion des champs chromatiques est assez difficile en raison du peu de qualité ou de la rareté des ressources linguistiques existantes dans ce domaine.

Nous proposons un processus automatique de rapprochements de couleurs par calcul de distances euclidiennes à partir d'informations chromatiques largement et librement accessibles sur le Web. Ces rapprochements sont directement utilisées afin de déterminer les réponses acceptables dans la base de données pour des requêtes portant sur les couleurs. Nous présentons les résultats obtenus à partir des informations chromatiques contenues dans Wikipédia.

12.1 Introduction

Les distances et similarités perceptibles entre couleurs sont complexes à déterminer de manière automatique. A titre d'exemple, nous pouvons citer les couleurs « rouge », « vermillon » et « lave ». Il est évident pour un humain qu'elles appartiennent à un même champ chromatique, mais il n'en est pas de même pour un processus automatique privé d'informations externes. L'utilisation d'informations linguistiques généralistes devrait aider à contourner cette difficulté mais il est également évident que la pertinence de la réponse fournie est hautement corrélée à la qualité et à la disponibilité des ressources. Ces pré-requis ne peuvent être garantis pour toutes les langues et particulièrement pour le français.

Dans les bases de données, les couleurs sont habituellement exprimées en langage naturel sous la forme de valeurs de type chaîne de caractères. Comme n'importe quelle information de ces bases, elles doivent pouvoir être utilisées pour extraire des n-uplets pertinents à partir d'une requête. Malheureusement, les couleurs ne peuvent être utilisées directement dans ce sens. Dans la majorité des cas, les différences d'expressions linguistiques entre les couleurs de la base de données et celles des requêtes confrontent au problème des relations entre couleurs vu précédemment. Les résultats retournés sont souvent drastiquement réduits rendant la recherche peu pertinente.

Dans la section suivante de ce chapitre, nous étudions les problèmes de recherche de couleurs dans les bases de données et les difficultés à définir un type couleur efficace. Nous montrons comment les relations entre couleurs peuvent aider à résoudre ces problèmes. La section 12.3 est dédiée à la présentation de transformations sur les modèles colorimétriques standard dans le but de résoudre les problèmes présentés auparavant lors du calcul de distance entre couleurs. Dans la section 12.4, nous présentons notre stratégie de calcul de relations entre les couleurs et la définition du type couleur à partir de celles-ci. Finalement, la dernière section du chapitre est dédiée à l'évaluation des relations calculées ainsi qu'à la capacité induite de la base de données à fournir des résultats probants pour une requête donnée.

12.2 Recherche de couleurs

12.2.1 Les problèmes de la recherche de couleurs dans une base de données

Soit un type de base de données correspondant à un type complexe pour décrire des couleurs dans une base. Nous pouvons le définir naïvement au travers d'une table simple dénotée « color » ne comportant qu'une colonne « text » pour la description en langage naturel de la couleur. Cette colonne « text » peut évidemment contenir les noms des couleurs afin d'être utilisée dans des requêtes. Le but est de définir un opérateur « like » efficace sur cette colonne.

Le domaine de définition de ce type couleur est large et ouvert. En fait, le domaine des couleurs est limité seulement par la richesse de la langue et du domaine d'application concernés. Les noms de couleur dans les langues sont principalement mais non exclusivement fondés sur la perception globale des entités physiques désignés par des termes. Comme exemple classique, la couleur « orange » est référée à la perception globale du fruit pareillement nommé. Des exemples moins classiques que celui-ci concernent les couleurs « lave » et « bleu Air Force » utilisées largement dans cette thèse. Ces faits induisent une nécessité d'évolutivité du type couleur pour être adapté à la fois à la langue et au domaine d'application.

Dans la plupart des cas, les données présentes dans les bases sont aussi spécifiques que possible. Néanmoins, dans le cadre de traitements automatiques de couleurs, il peut être difficile d'extraire des informations moins spécifiques de manière automatique. Par exemple, nous pouvons requérir une couleur spécifique en SQL :

```
1 select * from color where text like 'lave'
```

Si la base contient la couleur « lave », elle sera trouvée assez facilement. Mais dans le cas où la base contient « vermillon », avec une recherche textuelle stricte, nous ne pouvons pas le trouver. Pourtant, « vermillon » est une réponse intéressante à la requête.

Nous pouvons aussi rechercher une couleur « générale » ou « vague » :

```
1 select * from color where text like 'rouge'
```

Dans ce cas et avec la base de données de l'exemple précédent, aucune couleur n'est trouvée alors que « vermillon » et « lave » sont toujours des résultats intéressants. En fait, une telle recherche d'une couleur « générale » est reliée à la notion bien connue de champ chromatique. Il doit alors être supposé pour ce type de requêtes que toute couleur appartenant au champ chromatique est pertinente pour la requête.

Dans la plupart des cas, les couleurs requises sont vagues de par le fait de la non-information des utilisateurs sur les couleurs réellement présentes dans la base de données. Néanmoins, nous devons gérer tous les cas.

De plus, il devrait être possible de rechercher des couleurs en fonctions de catégories perceptuelles. Par exemple, nous devrions pouvoir chercher :

```
1 select * from color where text like 'sombre'
```

Aucune couleur ne peut être trouvée par sa catégorie perceptuelle avec la définition naïve du type couleur.

C'est pourquoi, la construction naïve du type couleur évoquée plus haut avec seulement une table contenant uniquement une colonne « text » n'est pas suffisante. Nous proposons à la place de construire une table couleur avec deux colonnes dénotées « searchedText » et « colorName ». Comme leurs noms le suggèrent, la première colonne contient les textes qu'il est possible de rechercher tandis que la seconde contient les noms des couleurs qui correspondent à la recherche. La construction de la table doit être automatique en raison du coût de construction et de maintenance de celle-ci.

12.2.2 État de l'art de la recherche de couleur dans une base de données

La recherche de couleurs n'est pas une problématique nouvelle. Le domaine d'application le plus important est celui des systèmes de recherche d'images par leur contenus. Dans ces systèmes, le but est de retrouver des images dans une base de données en fonction des couleurs qu'elles contiennent. Par exemple, dans (STANCHEV, GREEN JR et DIMITROV 2003), des peintures sont recherchées grâce à des propriétés colorimétriques qualifiées de haut niveau comme les contrastes blanc-noir

ou chaud-froid. Dans ce travail, le modèle CIE Luv est utilisé pour la représentation des couleurs. Dans (LI, SHI et LUO 2007), c'est le modèle TSV qui est utilisé.

Comme autres exemples relatifs à notre travail, nous pouvons citer les projets "*Name that color*" [1] et "*Name of colors*" [2]. Ces projets cherchent à identifier à partir de la définition d'une couleur non nommée, la plus proche couleur nommée équivalente dans une base de données. La problématique ciblée dans nos travaux est plus générale que celle du nommage, le but n'étant pas seulement de déterminer la couleur la plus proche mais bien de manière générale l'ensemble des couleurs acceptables pour une requête de couleur.

Des cas problématiques apparaissent avec le projet "*Name that color*" spécialement pour les couleurs achromatiques comme le blanc, le noir et les gris. Le blanc cassé est ainsi relié à un rose pâle alors qu'il semblerait préférable de le relier à du blanc (voir l'outil en ligne). Quelques sélections de noirs peuvent conduire à obtenir un marron profond plutôt qu'un noir (voir l'outil en ligne).

Ces cas problématiques n'apparaissent pas dans le projet "*Name of colors*". Il y a plus de 2000 définitions différentes de couleurs dans la base de données de ce projet. Il est possible que ces problèmes soient contournés par un nombre de définitions de couleurs important dans ces zones difficiles.

Comme notre problématique est plus générale que celle de déterminer la couleur nommée la plus proche, nous sommes confrontés à des difficultés similaires à celles rencontrées par les projets "*Name that color*" et "*Name of colors*". Ces difficultés ne peuvent pas être contournées par un nombre important de définitions de couleurs. En effet, nous voulons déterminer toutes les couleurs similaires dans l'espace et non pas juste la plus proche.

12.2.3 Recherche de couleurs dans les bases de données par dissimilarité colorimétrique

Il n'est pas évident de déterminer une dissimilarité colorimétrique idéale entre des couleurs même si cela est réalisé manuellement. En fait, nous postulons que cette

1. `http://chir.ag/projects/name-that-color`
2. `http://name-of-color.com`

notion de dissimilarité idéale est une notion plus subjective qu'objective. Prenons un exemple pour illustrer le phénomène. Prenons en français les couleurs distinctes Ambre, Jaune Mikado et Jaune sélectif. La table 12.1 donne des exemples de ces couleurs. Supposons que nous cherchons une dissimilarité idéale donnant la couleur la plus proche de Ambre. Est ce la dissimilarité donnant le Jaune Mikado comme couleur la plus proche ou bien celle donnant le Jaune sélectif comme couleur la plus proche que nous devons considérer comme idéale ?

En revanche, il est souvent possible d'établir une relation d'ordre entre les dissimilarités colorimétriques basée sur la qualité perceptible de la séparation engendrée sur les couleurs. Prenons la couleur Jaune pastel en plus des trois précédentes. La table 12.1 donne un exemple de ces couleurs. Cette fois, il est objectif de considérer que les dissimilarités donnant les couleurs Ambre et Jaune mikado et les couleurs Ambre et Jaune sélectif plus « proches » que les couleurs Ambre et Jaune pastel sont meilleures que les autres.

Même si une telle relation d'ordre peut souvent être établie, elle n'est ni aisée à déterminer, ni aisée à vérifier. Cela est principalement dû à une absence d'intuition pour les humains de la majeure partie des dissimilarités colorimétriques que nous pouvons établir. En revanche, les représentations colorimétriques présentées auparavant permettent d'utiliser la distance euclidienne, distance intuitive pour un humain. L'utilisation d'une telle distance est intéressante en matière de validation des résultats obtenus.

Dans ce cadre, plutôt que de chercher une dissimilarité correspondante à nos besoins de séparations des couleurs, nous cherchons un modèle colorimétrique compatible avec l'application d'une distance euclidienne et fournissant pour celle-ci une bonne séparation des couleurs. Nous établissons ainsi une relation d'ordre entre les représentations colorimétriques basée sur la qualité perceptible de la séparation euclidienne des couleurs.

L'opérateur « like » peut dès lors être défini selon cette distance euclidienne simplement en considérant un rayon de voisinage autour des couleurs acceptables en terme de perception homogène des couleurs. Nous discutons plus précisément des rayons de voisinage acceptables plus loin dans le chapitre.

Ambre	Jaune Mikado	Jaune sélectif	Jaune pastel

TABLE 12.1 – Exemples de couleurs similaires ou non à Ambre. Chaque colonne donne le nom de la couleur en 1re ligne et un échantillon de celle-ci en 2e ligne. Jaune Mikado et Jaune sélectif sont similaires à Ambre contrairement à Jaune pastel beaucoup plus clair.

12.3 Transformation de modèles colorimétriques

Les modèles colorimétriques existants ne sont pas parfaits pour notre tâche et ceci spécialement pour le blanc, le noir et les gris.

Les modèles TSL et TCL sont plus intéressants à utiliser que des modèles plus complexes comme CIE Lab car ils apparaissent plus intuitifs et plus simples à implémenter. Le but de ce travail est donc d'adapter ces modèles pour surpasser les problèmes évoqués tout en conservant dans le même temps la simplicité de ces modèles.

Pour surmonter les difficultés, nous appliquons des transformations aux modèles colorimétriques originaux dans le but de résoudre les problèmes locaux autour du blanc, noir et gris. Nous présentons des transformations dans les modèles TSL et TCL. Le but est de fournir un modèle colorimétrique dans lequel il est possible de calculer une distance euclidienne homogène entre les couleurs n'importe où dans celui-ci. Par la suite, la saturation pour la luminosité est appelée simplement saturation.

12.3.1 Transformation du modèle TSL

Nous définissons d'abord une transformation sur le modèle TSL par une application $\Phi^s_{n,m}$. Soient les couleurs définies dans le modèle TSL par $(h,s,l) \in [\![0,360[\![\times [0,1]^2$. Nous définissons l'application $\Phi^s_{n,m}$ avec $(n,m) \in \mathbb{R}^{+2}$.

$$\Phi^s_{n,m}: \quad [\![0,360[\![\times [0,1]^2 \rightarrow [-1,1]^2 \times [0,1]$$
$$\begin{pmatrix} h \\ s \\ l \end{pmatrix} \mapsto \begin{pmatrix} x' = s^n \cos h \\ y' = s^n \sin h \\ l' \end{pmatrix}$$

$avec$:

$$l' = \begin{cases} \frac{(2\times l)^m}{2} & \text{if } l < 0.5 \\ 1 - \frac{(2(1-l))^m}{2} & \text{if } l \geq 0.5 \end{cases}$$

Dans l'application $\Phi^s_{n,m}$ définie ci-dessus, les trois attributs naturels du modèle TSL sont transformés en trois nouveaux attributs x', y' et l'. Les deux premiers sont corrélés car ils correspondent à une transformation des coordonnées polaires formées par la Teinte et la Saturation en coordonnées cartésiennes dans $[-1,1]^2$. Le troisième et dernier correspond à une transformation de la Luminosité.

12.3.2 Transformation du modèle TCL

Nous définissons ensuite une transformation sur le modèle TCL par l'application $\Phi^c_{n,m}$. Soient les couleurs définies dans le modèle TCL par $(h,c,l) \in [\![0,360[\![\times [0,1]^2$. Nous définissons l'application $\Phi^c_{n,m}$ avec $(n,m) \in \mathbb{R}^{+2}$.

$$\Phi^c_{n,m}: \quad [\![0,360[\![\times [0,1]^2 \rightarrow [-1,1]^2 \times [0,1]$$
$$\begin{pmatrix} h \\ c \\ l \end{pmatrix} \mapsto \begin{pmatrix} x'' = c^n \cos h \\ y'' = c^n \sin h \\ l' \end{pmatrix}$$

Dans l'application $\Phi^c_{n,m}$ définie ci-dessus, les trois attributs du modèle TCL sont transformés en trois nouveaux attributs x'', y'' et l'. Les deux premiers sont corrélés et correspondent à une transformation des coordonnées polaires formées par la Teinte et la Chroma en coordonnées cartésiennes dans $[-1,1]^2$. Le troisième et dernier correspond à une transformation de la Luminosité.

12.3.3 Discussions sur les transformations

Les applications $\Phi^s_{n,m}$ et $\Phi^c_{n,m}$ présentées auparavant sont paramétrées par deux paramètres dénotés respectivement n et m. Ils adaptent la transformation du modèle colorimétrique dans le but de donner une perception homogène de la distance euclidienne en tous points de l'espace colorimétrique. Le paramètre n agit comme un ratio appliqué à la Saturation (resp. Chroma), rayon des coordonnées polaires avant la conversion cartésienne. Le paramètre m agit comme un ratio appliqué sur la Luminosité. L'application $\Phi^s_{n,m}$ transforme la Luminosité de la même manière que l'application $\Phi^c_{n,m}$. Les deux applications partagent leur dernier attribut l'.

Il est à noter que quand le paramètre n vaut 1, le rayon des coordonnées polaires formées par la Teinte et la Saturation (resp. la Teinte et la Chroma) n'est pas modifié avant la conversion cartésienne. De même, quand le paramètre m vaut 1, la Luminosité n'est pas transformée. Cela induit que lorsque les paramètres valent 1, aucune transformation n'est appliquée au modèle colorimétrique original.

La figure 12.1 donne la transformation $\Phi^s_{n,m}$ (resp. $\Phi^c_{n,m}$) de la valeur de s (resp. c) en la valeur s^n (resp. c^n) pour différentes valeurs du paramètre n.

La figure 12.2 donne les transformations $\Phi^s_{n,m}$ et $\Phi^c_{n,m}$ de la valeur l en la valeur l' pour différentes valeurs du paramètre m.

12.4 Distances dans les modèles colorimétriques transformés

Grâce aux applications définies au préalable, des distances peuvent être calculées dans les modèles colorimétriques résultants. Comme nous l'avons indiqué à la section 12.1, nous cherchons à construire une table de relations entre les couleurs et les recherches qui peuvent être des couleurs ou des catégories perceptuelles. La table est donnée par la relation suivante :

$$color(searchedText, colorName)$$

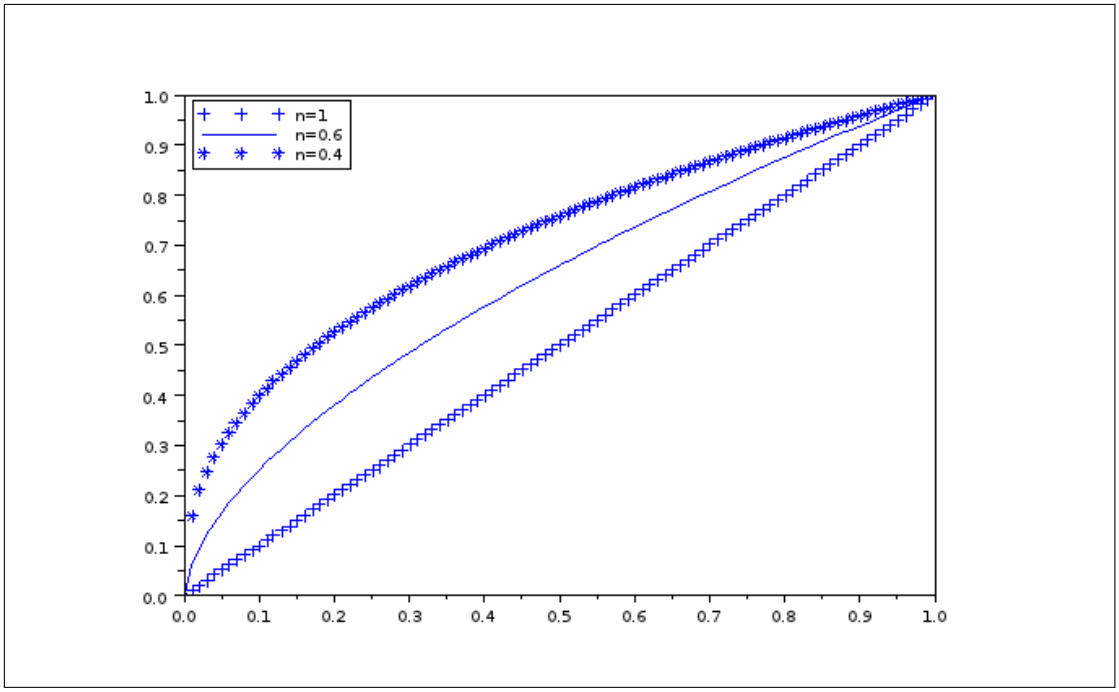


FIGURE 12.1 – Rayon s^n (resp. c^n) utilisé dans les coordonnées polaires en tant que fonction de s (resp. c) pour différentes valeurs du paramètre n

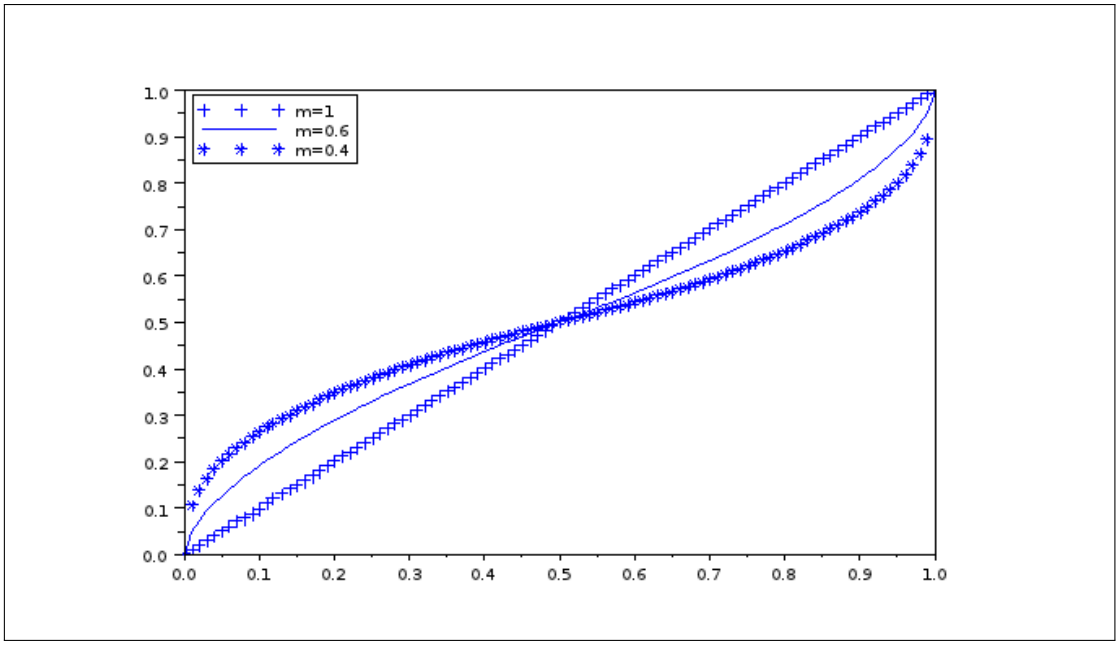


FIGURE 12.2 – Luminosité transformée l' en tant que fonction de l pour différentes valeurs du paramètre m

12.4.1 Distance euclidienne entre deux couleurs

Soient deux couleurs définies dans les modèles TSL et TCL par les vecteurs d'attributs $\vec{c_1^s}$, $\vec{c_2^s}$, $\vec{c_1^c}$ et $\vec{c_2^c}$. Nous définissons une distance euclidienne entre couleurs dans le modèle colorimétrique grâce aux attributs du modèle transformé. Soient $Rel^s_{n,m}$ et $Rel^c_{n,m}$, les relations entre deux couleurs dans les modèles TSL et TCL transformés.

$$Rel^s_{n,m}(\vec{c_1^s}, \vec{c_2^s}) = \frac{||\Phi^s_{n,m}(\vec{c_1^s}) - \Phi^s_{n,m}(\vec{c_2^s})||}{\sqrt{5}}$$
$$Rel^c_{n,m}(\vec{c_1^c}, \vec{c_2^c}) = \frac{||\Phi^c_{n,m}(\vec{c_1^c}) - \Phi^c_{n,m}(\vec{c_2^c})||}{2}$$

Une normalisation de la distance est appliquée en accord avec la distance maximale calculable dans le modèle colorimétrique.

12.4.2 Distance euclidienne entre les couleurs et les champs chromatiques

Nous pouvons établir une liste de champs chromatiques pour un langage déterminé. Par exemple, pour le français, pour l'anglais et plus généralement pour toutes les langues ayant atteinte la septième étape de l'évolution des champs chromatiques (BERLIN et KAY 1969), il y a onze champs chromatiques : rose, violet, bleu, vert, jaune, marron, orange, rouge, noir, blanc et gris. Ces champs sont en fait dénotés par la couleur occupant le centre du champ chromatique. Ces centres sont bien sûr des couleurs standard possédant comme les autres une position dans le modèle colorimétrique.

Une conséquence directe est la possibilité de calculer des relations entre n'importe quelle couleur et les centres des champs chromatiques. De plus, nous pouvons supposer que chaque couleur est plus proche du centre de son champ chromatique que d'un autre.

Cependant, ce postulat n'est pas vérifié pour les couleurs proches du cyan. Cyan est une couleur située entre le bleu et le vert où il n'y a aucun champ chromatique. Nous contournons ce problème en créant un champ chromatique fictif cyan. Si l'on

ne souhaite pas conserver ce champ a posteriori, lorsqu'il est sélectionné par notre procédure, la couleur peut être attribuée aux deux champs chromatiques du vert et du bleu.

Soient les vecteurs $\vec{c^s}$ et $\vec{c^c}$, représentatifs d'une couleur définie dans les modèles TSL et TCL. Soient les vecteurs $\vec{cf^s}$ et $\vec{cf^c}$, représentatifs d'une couleur d'un centre de champ dans les modèles TSL et TCL. Nous définissons une distance euclidienne entre une couleur et les champs chromatiques dans un modèle transformé grâce aux attributs du modèle transformé appliqué sur la couleur et les centres. Soient $RelCf^s_{n,m}$ et $RelCf^c_{n,m}$, les relations entre une couleur et les champs chromatiques des modèles TSL et TCL transformés. On pose :

$$RelCf^s_{n,m}(\vec{c^s}) = \arg\min_{\vec{cf^s}} Rel^s_{n,m}(\vec{c^s}, \vec{cf^s})$$
$$RelCf^c_{n,m}(\vec{c^c}) = \arg\min_{\vec{cf^c}} Rel^c_{n,m}(\vec{c^c}, \vec{cf^c})$$

12.4.3 Distance de Manhattan entre les couleurs et les catégories perceptuelles

La distance de Manhattan est, après la distance euclidienne, la distance la plus intuitive pour les humains. Nous pouvons dès lors considérer qu'elle offre les mêmes facilités de validation que la distance euclidienne. Elle est de surcroît souvent utilisée pour identifier des catégories perceptuelles avec le modèle TSL.

Nous postulons donc qu'il est possible de l'utiliser à cette même fin dans les modèles colorimétriques transformés. La table 12.2 donne les catégories correspondantes pour une couleur c connaissant les valeurs s^n (resp. c^n) et l'. Soient $RelCat^s_{n,m}$ et $RelCat^c_{n,m}$, les applications donnant pour une couleur donnée sa catégorie perceptuelle dans les modèles TSL et TCL transformés. La table 12.2 donne le résultat retourné par ces application selon les valeurs de s, c, l', n et m.

$RelCat^s_{n,m}$ ou $RelCat^c_{n,m}$	$s^n < \frac{1}{3}$ ou $c^n < \frac{1}{3}$	$\frac{1}{3} < s^n < \frac{2}{3}$ ou $\frac{1}{3} < c^n < \frac{2}{3}$	$s^n > \frac{2}{3}$ ou $c^n > \frac{2}{3}$
$l' > \frac{2}{3}$	pâle	claire	lumineuse
$\frac{1}{3} < l' < \frac{2}{3}$	délavée	moyenne	vive
$l' < \frac{1}{3}$	sombre	foncée	profonde

TABLE 12.2 – Catégories perceptuelles en fonction du rayon s^n (resp. c^n) et des valeurs de l'. Les colonnes donnent les plages de valeurs du rayon s^n (resp. c^n). Les lignes donnent les plages de valeurs de l'. L'intersection des lignes et des colonnes donne le nom de la catégorie perceptuelle retournée par l'application $RelCat^s_{n,m}$ (resp. $RelCat^c_{n,m}$) ainsi qu'un échantillon de couleur proche du bleu souhaitable dans cette catégorie.

12.5 Construction de la table couleur

La table couleur est construite par un processus automatique décrit par les algorithmes 3, 4, 5, 6, 7, 8 et 9.

Ce processus prend cinq paramètres en entrée. Soit la table $incolor$ contenant les noms et les définitions chromatiques définie par : $incolor(name, h, s, l, c)$. Soient les paramètres complexes R, Cf et Cat contenant chacun les trois paramètres $model$, n et m, chaînes de caractères indiquant si la table doit être construite depuis les modèles $'hsl'$ ou $'hcl'$ et avec quelle transformation et ce pour les trois types de relations. Soit finalement le paramètre $threshold$, une valeur numérique comprise dans $[0,1]$ qui indique le seuil désiré pour les relations entre couleurs. La table couleur est remplie selon ces données et les distances euclidiennes et de Manhattan présentées auparavant.

Input : $incolor, n, m, threshold$
Output : nothing

```
1  begin
2      forall the c1 ∈ incolor do
3          c1⃗ ← (c1.h, c1.s, c1.l) ;
4          forall the c2 ∈ incolor do
5              c2⃗ ← (c2.h, c2.s, c2.l) ;
6              if Rel^s_{n,m}(c1⃗, c2⃗) < threshold then
7                  color ← color ∪ (c1.name, c2.name) ;
8              end
9          end
10     end
11 end
```

Algorithme 3 : Algorithme de génération des relations colorimétriques depuis TSL : La table couleur est remplie avec les associations de couleurs dont la distance euclidienne est inférieure au rayon de voisinage considéré comme homogène en perception selon le modèle TSL transformé

Input : $incolor, n, m$
Output : nothing

```
1  begin
2      forall the c1 ∈ incolor do
3          c1⃗ ← (c1.h, c1.s, c1.l) ;
4          color ← color ∪ (RelCf^s_{n,m}(c1⃗), c1.name) ;
5      end
6  end
```

Algorithme 4 : Algorithme de génération des champs chromatiques depuis TSL : La table couleur est remplie avec les associations champ chromatique / couleurs dont la distance euclidienne est minimale par rapport à toutes les couleurs centroïdes d'un champ chromatique selon le modèle TSL transformé

Input : $incolor, n, m$
Output : nothing
begin
  forall the $c1 \in incolor$ **do**
    $\vec{c1} \leftarrow (c1.h, c1.s, c1.l)$;
    $color \leftarrow color \cup (RelCat^{s}_{n,m}(\vec{c1}), c1.name)$;
  end
end

Algorithme 5 : Algorithme de génération des catégories perceptuelles depuis TSL : La table couleur est remplie avec les associations catégorie perceptuelle / couleurs dont la distance de Manhattan correspond selon le modèle TSL transformé

Input : $incolor, n, m, threshold$
Output : nothing
begin
  forall the $c1 \in incolor$ **do**
    $\vec{c1} \leftarrow (c1.h, c1.c, c1.l)$;
    forall the $c2 \in incolor$ **do**
      $\vec{c2} \leftarrow (c2.h, c2.c, c2.l)$;
      if $Rel^{c}_{n,m}(\vec{c1}, \vec{c2}) < threshold$ **then**
        $color \leftarrow color \cup (c1.name, c2.name)$;
      end
    end
  end
end

Algorithme 6 : Algorithme de génération des relations colorimétriques depuis TCL : : La table couleur est remplie avec les associations de couleurs dont la distance euclidienne est inférieure au rayon de voisinage considéré comme homogène en perception selon le modèle TCL transformé

Input : $incolor, n, m$
Output : nothing
begin
  forall the $c1 \in incolor$ **do**
    $\vec{c1} \leftarrow (c1.h, c1.c, c1.l)$;
    $color \leftarrow color \cup (RelCf^{c}_{n,m}(\vec{c1}), c1.name)$;
  end
end

Algorithme 7 : Algorithme de génération des champs chromatiques depuis TCL : La table couleur est remplie avec les associations champs chromatiques / couleurs dont la distance euclidienne est minimale par rapport à toutes les couleurs centroïdes d'un champ chromatique selon le modèle TCL transformé

Input : $incolor, n, m$
Output : nothing
begin
  forall the $c1 \in incolor$ **do**
    $\vec{c1} \leftarrow (c1.h, c1.c, c1.l)$;
    $color \leftarrow color \cup (RelCat^{c}_{n,m}(\vec{c1}), c1.name)$;
  end
end

Algorithme 8 : Algorithme de génération des catégories perceptuelles depuis TCL : La table couleur est remplie avec les associations catégorie perceptuelle / couleurs dont la distance de Manhattan correspond selon le modèle TCL transformé

Input : $incolor, R, Cf, Cat, threshold$
Output : nothing

```
1  begin
2      color ← {} ;
3      if R.model =′ hsl′ then
4          fillRFromHsl(incolor, R.n, R.m, threshold) ;
5      end
6      else
7          if R.model =′ hcl′ then
8              fillRFromHcl(incolor, R.n, R.m, threshold) ;
9          end
10     end
11     if Cf.model =′ hsl′ then
12         fillCfFromHsl(incolor, Cf.n, Cf.m) ;
13     end
14     else
15         if Cf.model =′ hcl′ then
16             fillCfFromHcl(incolor, Cf.n, Cf.m) ;
17         end
18     end
19     if Cat.model =′ hsl′ then
20         fillCatFromHsl(incolor, Cat.n, Cat.m) ;
21     end
22     else
23         if Cat.model =′ hcl′ then
24             fillCatFromHcl(incolor, Cat.n, Cat.m) ;
25         end
26     end
27 end
```

Algorithme 9 : Algorithme de génération de la table des couleurs : La table couleur est remplie successivement avec la distance euclidienne entre couleurs, avec la distance euclidienne entre couleurs et champs chromatiques et avec la distance de Manhattan des catégories perceptuelles et ceci que l'on travaille avec le modèle TSL ou TCL

12.6 Expérimentations

12.6.1 Sources d'informations chromatiques

De nombreuses sources d'informations chromatiques sont librement accessibles sur le Web. Wikipédia est l'une des plus intéressantes en raison d'une grande communauté de contributeurs. Les pages listant des couleurs donnent des informations chromatiques sur des couleurs de nombreux langages, dont environ 800 couleurs en anglais [3] et 400 couleurs en français [4] [5]. Les propriétés RVB sont toujours renseignées alors que les propriétés TSV et TSL sont fournies plus rarement. De plus, aucune d'elles ne propose la propriété Chroma. Le manque de ces informations n'est pas un problème puisqu'elles correspondent à des transformations du modèle RVB. De surcroît, quand elles sont fournies, elles ne sont pas forcément cohérentes avec la transformation des propriétés RVB. Par exemple, sur la version française, les propriétés TSL données pour la couleur Ambre sont $(49, 1.0, 0.94)$. Ces informations sont erronées car en accord avec les propriétés RVB données sur la même page, les bonnes informations TSL sont $(49, 1.0, 0.47)$. Le mauvais n-uplet fourni correspond aux propriétés TSV pour ambre. C'est pourquoi nous extrayons seulement les propriétés RVB de Wikipédia et calculons les propriétés de TSL et TCL à partir de celles-ci.

Il y a d'autres sites Internet qui proposent des définitions chromatiques de couleurs. Par exemple, nous pouvons citer pourpre.com qui donne 522 définitions de couleurs en français [6] ou le dictionnaire colorimétrique de code-couleur.com qui donne aussi des définitions de couleurs en français [7].

Comme les informations chromatiques sont extraites du Web, nous supposons raisonnablement travailler dans l'espace sRVB.

3. `http://en.wikipedia.org/wiki/List_of_colors`
4. `http://fr.wikipedia.org/wiki/Liste_de_couleurs`
5. Données consultées et extraites le 4 mars 2012
6. `http://pourpre.com/chroma/zip.php`
7. `http://www.code-couleur.com/dictionnaire/index.html`

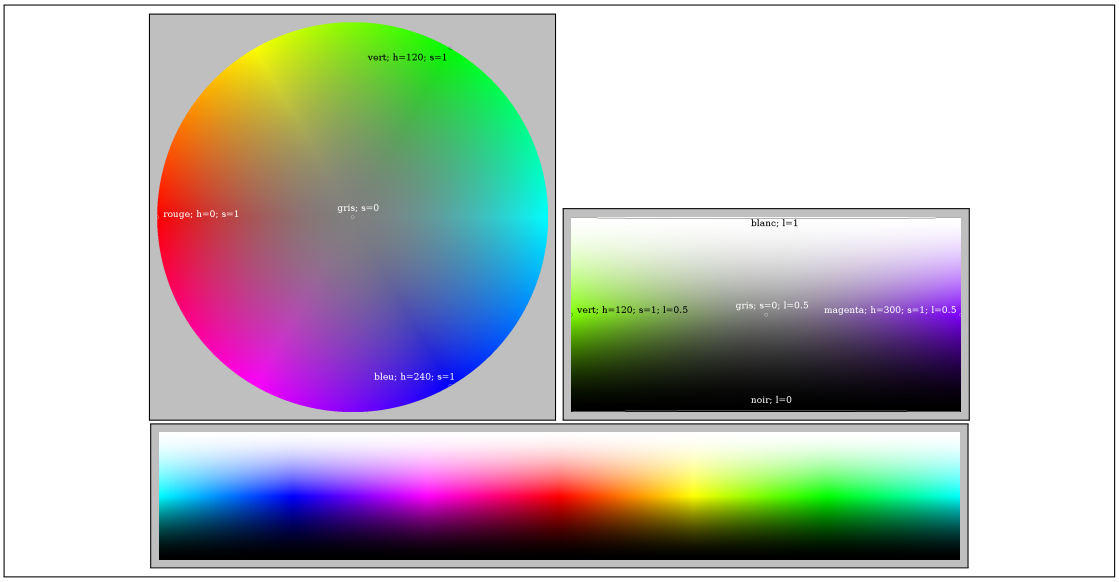


FIGURE 12.3 – Coupes de l'espace colorimétrique sRVB avec le modèle TSL transformé par $\Phi^{s}_{0.6,0.6}$. Une coupe horizontale, verticale et circulaire sont présentées de gauche à droite et de haut en bas.

12.6.2 Influence des valeurs de paramètres sur les modèles colorimétriques transformés

Le but est de déterminer les meilleures valeurs pour les paramètres n et m pour contourner les problèmes présentés auparavant. Comme nous en avons discuté, sélectionner les valeurs 1 et 1 pour les paramètres résulte en l'utilisation du modèle d'origine alors que l'emploi de valeurs inférieures à 1 résulte en l'emploi d'un modèle transformé où le noir, le blanc et les gris sont affectés.

Dans un premier temps, il est raisonnable d'essayer une transformation symétrique limitée afin d'affecter raisonnablement et uniformément les couleurs problématiques. Soient les transformations $\Phi^{s}_{0.6,0.6}$ et $\Phi^{c}_{0.6,0.6}$ sur les modèles TSL et TCL. Les figures 12.3 et 12.4 donnent des coupes de ces modèles colorimétriques transformés.

Dans ces premiers modèles transformés, nous pouvons observer que les zones du blanc, du noir et des gris ont des surfaces plus grandes que dans les modèles originaux. Grâce à la transformation, ces zones apparaissent plus équilibrées avec les autres couleurs que dans les modèles originaux. Néanmoins, les zones noires et blanches semblent être un peu plus petites que les zones vertes ou magenta.

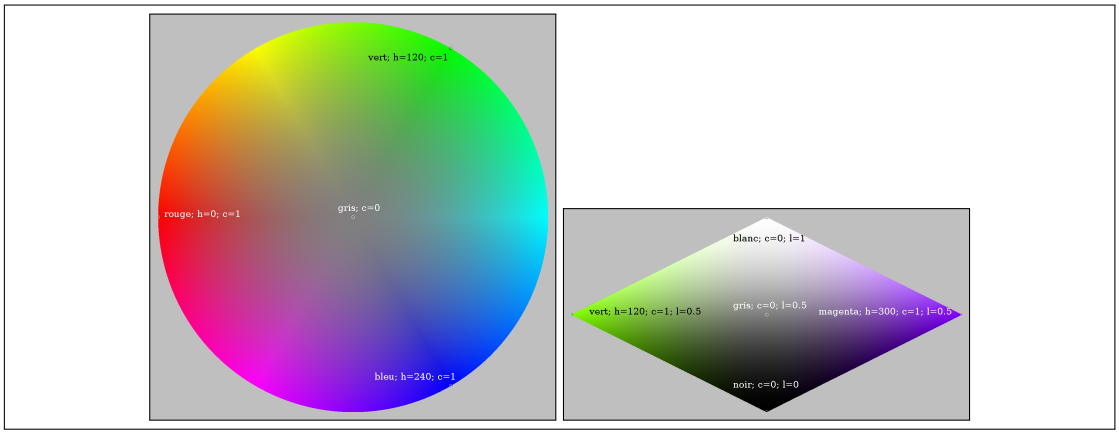


FIGURE 12.4 – Coupes de l'espace colorimétrique sRVB avec le modèle TCL transformé par $\Phi^{c}_{0.6,0.6}$. Une coupe horizontale et verticale sont présentées de gauche à droite.

C'est pourquoi, dans un second temps, il semble raisonnable d'essayer une transformation asymétrique dans le but d'affecter plus sensiblement le blanc et le noir que le gris. Soient les transformations $\Phi^{s}_{0.6,0.4}$ et $\Phi^{c}_{0.6,0.4}$ appliquées sur les modèles TSL et TCL. Les figures 12.5 et 12.6 donnent les coupes de ces modèles transformés.

Dans ces seconds modèles transformés, les zones noires et blanches apparaissent aussi larges que les zones colorées.

Les évaluations suivantes sont réalisées sur les modèles transformés obtenus à partir des deux ensembles de paramètres sélectionnés dans cette sous-section.

12.6.3 Évaluation des distances dans les modèles colorimétriques transformées

Nécessité d'un outil de test

Dans le but de comparer et évaluer les distances trouvées dans les différents modèles colorimétriques, un outil de test a été développé en tant qu'application Web disponible en ligne [8]. Il est ainsi possible pour le lecteur de se forger sa propre opinion sur l'adéquation entre proximité perceptuelle des couleurs et distance entre celles-ci dans l'espace de représentation.

8. `http://labs.onyme.com/`

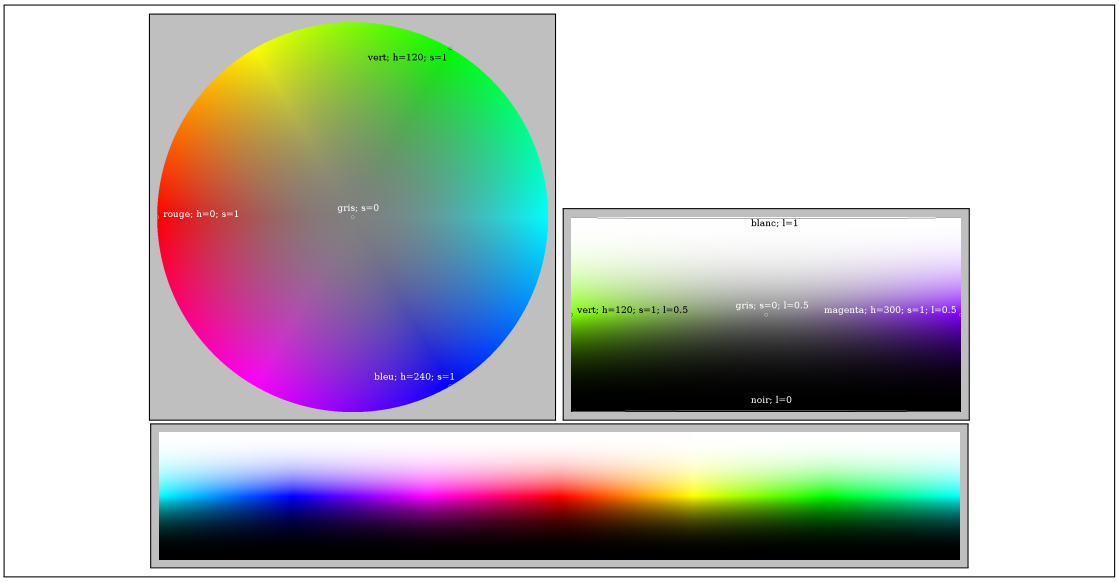


FIGURE 12.5 – Coupes de l'espace colorimétrique sRVB avec le modèle TSL transformé par $\Phi^{s}_{0.6,0.4}$. Une coupe horizontale, verticale et circulaire sont présentées de gauche à droite et de haut en bas.

Il est possible avec cet outil de rechercher les relations entre deux couleurs, les relations entre un champ chromatique et les couleurs qu'il contient et finalement les relations entre les catégories perceptuelles et les couleurs acceptables pour celles-ci.

Le seuil choisi pour l'affichage d'une relation entre deux couleurs est maximal (1.0) laissant ainsi toutes les relations calculables visibles. Il est bien sûr évident que les relations mettant en jeu les couleurs les plus éloignées ne sont que très peu pertinentes.

Ces informations sont disponibles pour les deux modèles de base TSL et TCL avec les valeurs de paramètres de transformation choisies dynamiquement. De plus, l'outil de test peut calculer des relations pour les couleurs extraites en français et en anglais de Wikipédia.

Par défaut, les relations sont calculées à partir du modèle TSL transformé par $\Phi^{s}_{0.6,0.4}$.

Dans l'ensemble des observations, nous utilisons régulièrement des couleurs pour exprimer notre point de vue sur les modèles colorimétriques considérés. La table 12.3 donne des exemples de ces couleurs. Elles correspondent à des traductions fran-

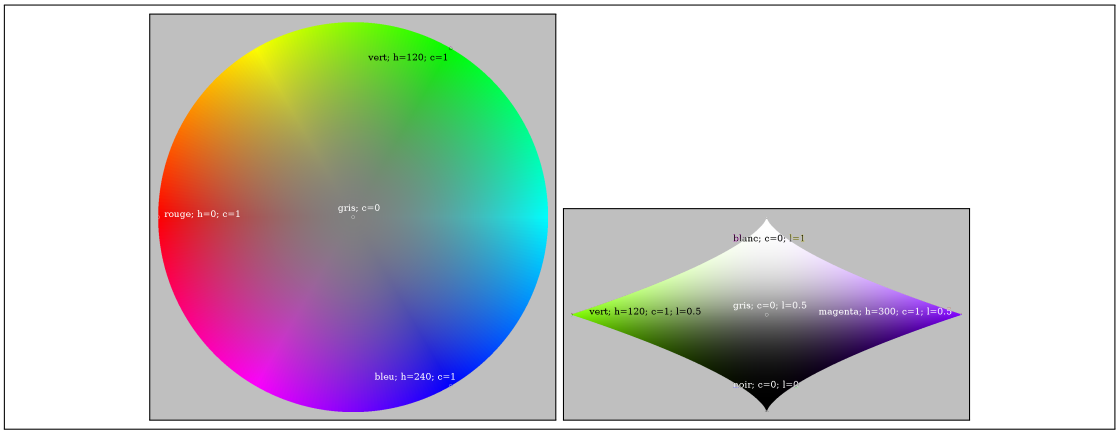


FIGURE 12.6 – Coupes de l'espace colorimétrique sRVB avec le modèle TCL transformé par $\Phi^{c}_{0.6,0.4}$. Une coupe horizontale et verticale sont présentées de gauche à droite.

çaises des couleurs anglaises renseignées entre parenthèses. Les observations faites par la suite sont issues des définitions anglaises.

Observations des distances dans les modèles colorimétriques de base

Les observations sont faites dans cette sous-section en prenant les valeurs de paramètres à 1 et 1.

Observations des distances dans le modèle TSL

Sélectionnons le modèle TSL comme modèle de base dans l'outil de test.

Nous recherchons les relations entre le blanc et les autres couleurs. Il y a peu de couleurs ayant beaucoup de relations avec le blanc. Nous avons seulement sept couleurs avec une distance plus petite que 0.1 avec le blanc. De plus, la plupart de ces couleurs sont plus proches du gris que de blanc. Malheureusement, il y a d'autres couleurs dans la base de données que nous relions plus naturellement et de manière plus forte au blanc que les gris. La couleur neige fait typiquement partie de celles-ci. Cette couleur apparaît dans le milieu de la liste avec une distance de 0.45 par rapport au blanc. Pire, nous pouvons voir que la couleur noire est légèrement plus proche de blanc que la couleur neige. Clairement, un modèle accordant une plus grande proximité entre blanc et neige qu'entre blanc et noir est meilleur que TSL sur ce point.

blanc (white)	neige (snow)
blanc fantôme (ghost white)	blanc fumé (white smoke)
gris (gray)	gris foncé (dark gray)
gris léger (light gray)	bleu Air Force (Air Force blue)
vert foncé (dark green)	pourpre taupe (purple taupe)
iceberg	perle d'eau (pearl aqua)
bleu de Caroline (Carolina blue)	céruléen pâle (pale cerulean)
Onyx	vert sève (sap green)
vert chasseur (hunter green)	
noir (black)	noir fumé (smoky black)

TABLE 12.3 – Exemples de couleurs utiles pour nos observations. Les couleurs que nous jugeons « semblables » sont regroupées ensemble dans le tableau.

Des observations similaires peuvent être faites avec les relations entre noir et les autres couleurs. Cette fois, le noir est moins similaire à vert foncé qu'au blanc fumé. Il y a une distance de 0.43 entre le noir et le vert foncé et seulement 0.42 entre le noir et le blanc fumé. Dans la même idée, le noir fumé est évalué à une distance de 0.15 du noir le laissant moins similaire au noir que le pourpre taupe par exemple.

Sélectionnons maintenant le champ chromatique du gris. Beaucoup de couleurs sont trouvés. Malheureusement, certaines d'entre elles ne sont pas relatives au champ chromatique du gris et, spécialement celles trouvées à une distance supérieure à 0.2 du gris. Par exemple, iceberg, perle d'eau, bleu de Caroline et céruléen pâle sont des couleurs qui ne sont pas reliées au champ chromatique du gris mais à celui du bleu.

Sélectionnons maintenant la catégorie perceptuelle sombre. Il y a quelques couleurs dans cette catégorie perceptuelle. Elle contient des couleurs vraiment sombres comme noir, Onyx ou noir fumé mais aussi des couleurs moins sombres comme par exemple le vert chasseur.

Observations des distances dans le modèle TCL

Sélectionnons maintenant le modèle TCL comme modèle de base dans l'outil de test.

Premièrement, les relations colorimétriques pour le blanc sont meilleures que celles trouvées pour le modèle TSL. Cela est basé sur l'observation d'une distance plus faible entre blanc et neige que dans le modèle précédent. Dans TCL, la distance entre les deux couleurs est seulement de 0.01. C'est la meilleure relation trouvée pour le blanc juste avant le blanc fantôme et le blanc fumé. De plus, il y a cette fois beaucoup de couleurs fortement liées au blanc. Il y a ainsi 44 couleurs avec une distance inférieure à 0.1 du blanc.

Deuxièmement, les relations colorimétriques pour le noir sont également meilleures que dans le modèle TSL. La distance entre le noir fumé et le noir est de 0.02. C'est la meilleure distance pour le noir. De plus, la distance avec vert foncé est gratifiée d'une meilleure distance que dans TSL avec une valeur de 0.1 qui la place au 10^e rang.

Il y a toujours beaucoup de couleurs pâles ou délavées plus proches du gris que, par exemple, gris foncé qui apparaît pourtant perceptuellement comme plus proche.

Sélectionnons maintenant le champ chromatique du gris. Il y a toujours beaucoup de couleurs trouvées et même plus que dans le modèle TSL. Comme dans TSL, certaines d'entre elles ne sont pas relatives au champ chromatique du gris, spécialement les couleurs trouvées par l'outil à une distance supérieure à 0.15 du gris. Les mêmes cas problématiques que précédemment tels que iceberg, perle d'eau, bleu de Caroline et céruléen pâle apparaissent. D'une manière générale, ces couleurs problématiques semblent être évaluées plus proches du gris dans le modèle TCL que dans TSL. Par exemple, le bleu de Caroline est évalué à une distance de 0.17 pour le modèle TCL contre 0.24 pour TSL.

Sélectionnons maintenant la catégorie perceptuelle sombre. Il y a plus de couleurs dans cette catégorie perceptuelle qu'avec le modèle TSL. Elle contient toujours des couleurs réellement sombres comme noir, Onyx ou noir fumé mais aussi des couleurs moins sombres comme par exemple le vert chasseur (comme dans le modèle TSL) ou le vert sève. D'une manière générale, le problème semble plus important dans le modèle TCL que dans le modèle TSL.

Observations des distances dans les modèles colorimétriques transformés

Les observations de cette sous-section sont obtenues en prenant les paramètres n et m inférieurs à 1.

Observations des distances dans le modèle transformé TSL

Sélectionnons le modèle TSL comme modèle de base avec une transformation symétrique par $\Phi^{c}_{0.6,0.6}$ dans l'outil de test.

Les observations montrent que la transformation ne résout pas les problèmes énoncés concernant les relations entre le blanc et les autres couleurs, comparé au modèle original. Pire, cette fois, il y a seulement une couleur qui se situe à une distance inférieure à 0.1 du blanc (blanc fumé).

Des observations similaires à celles faites sur le modèle original peuvent être réalisées avec les relations entre noir et les autres couleurs. Pire, la transformation provoque à nouveau des problèmes plus importants avec, par exemple, une distance de 0.23 entre le noir et le noir fumé.

Sélectionnons le modèle TSL comme modèle de base avec une transformation asymétrique par $\Phi^{c}_{0.6,0.4}$ dans l'outil de test.

Des observations similaires à celles faites pour la transformation précédente peuvent être faites sur les relations entre le blanc et les autres couleurs et le noir et les autres couleurs.

Concernant le champ chromatique du gris, aucun des 4 cas problématiques énoncés avec les modèles TSL et TCL ne sont présents dans ce modèle. Néanmoins, il y a toujours quelques problèmes avec les couleurs évaluées à une distance supérieure de 0.15 du gris telles que le bleu Air Force évalué à une distance de 0.21.

En revanche, concernant le champ chromatique du noir, seules les couleurs noir et Onyx y figurent. Le noir fumé n'y est donc pas comme l'on pourrait s'y attendre.

Sélectionnons maintenant la catégorie perceptuelle sombre. Il y a radicalement moins de couleurs sombres que dans le modèle TSL. La transformation appliquée réduit cette catégorie perceptuelle aux seules couleurs noire et Onyx. Cette réduction est typiquement trop drastique puisqu'elle supprime par exemple le noir fumé qui est une couleur attendue pour cette catégorie.

Observations des distances dans le modèle transformé TCL

Sélectionnons maintenant le modèle TCL en modèle de base avec une transformation symétrique par $\Phi^{c}_{0.6,0.6}$ dans l'outil de test.

Les relations colorimétriques pour le gris sont meilleures que dans le modèle original. L'ensemble des 9 meilleures couleurs trouvées sont des nuances de gris. Par exemple, le gris foncé est à la 5ème position avec une distance de 0.05. Cependant, il y a toujours quelques problèmes que nous pouvons essayer de surmonter. Par exemple, il y a le gris léger qui est évalué à une distance de 0.11 et est placé après beaucoup de couleurs pâles.

Sélectionnons maintenant le modèle TCL comme modèle de base avec une transformation asymétrique par $\Phi^{c}_{0.6,0.4}$ dans l'outil de test.

Les relations colorimétriques pour le gris sont légèrement meilleures que dans la transformation précédente. Nous avons toujours les 9 meilleures relations colorimétriques trouvées qui sont des nuances de gris. De plus, la relation entre le gris léger

est mieux évaluée avec une distance de 0.08 et un classement qui le place à la 12ème position.

Sélectionnons maintenant le champ chromatique du gris. Il y a à nouveau beaucoup de couleurs qui sont trouvées et donc plus qu'avec le modèle TSL en utilisant la même transformation. Comme dans le modèle TSL, certaines d'entre elles ne sont pas réellement reliées au champ chromatique du gris spécialement les couleurs trouvées par l'outil à une distance supérieure à 0.15 du gris. Les mêmes cas problématiques que précédemment tels que iceberg, perle d'eau, bleu de Caroline, céruléen pâle et bleu Air Force apparaissent. D'une manière générale, ces couleurs problématiques semblent être évaluées plus loin du gris dans le modèle TCL transformé que dans le modèle TCL. Par exemple, le bleu de Caroline est évalué à une distance de 0.17 du gris pour le modèle TCL contre une distance de 0.23 du gris pour le modèle TCL transformé.

En revanche, concernant le champ chromatique du noir, nous retrouvons cette fois les trois couleurs noir, Onyx et noir fumé. D'une manière générale, le modèle TCL semble meilleur que le modèle TSL sauf pour les champs du gris et du marron.

Sélectionnons maintenant la catégorie perceptuelle sombre. Il y a radicalement moins de couleurs que dans le modèle TCL et légèrement moins que dans le modèle TSL. La transformation appliquée réduit le nombre de couleurs attribuées à cette catégorie par le modèle TCL le plaçant à un niveau comparable au modèle TSL. De plus, les couleurs noir, noir fumé et Onyx sont toujours toutes dans la catégorie alors que les couleurs problématiques du modèle TSL comme le vert chasseur ou le vert sève sont sorties de la catégorie grâce à la transformation.

12.6.4 Construction et requêtes de la table couleur

Selon les observations faites et les conclusions tirées, la table couleur devrait être construite en prenant les paramètres n=0.6 et m=0.4 pour les trois types de relations. Le modèle TCL est choisi pour les relations entre couleurs, les catégories perceptuelles et les champs chromatiques autres que ceux du gris et du marron. Le modèle TSL étant choisi pour ces derniers. Le seuil pour les relations doit, quant à lui, être pris à 0.1 pour limiter les résultats erronés.

La recherche de lave (lava) (select colorName from color where searchedText like 'lava') donne 58 résultats dont Lava, Venetian red, Alizarin crimson, Rose madder, Harvard crimson, Fire engine red, Deep carmine pink, Red (pigment), Lust et Boston University Red comme top 10. Toutes les distances du top 10 sont inférieures à 0.05.

La recherche de rouge (red) (select colorName from color where searchedText like 'red') renvoie 81 couleurs dont Red, Candy apple red, KU Crimson, Red (RYB), Rosso corsa, Red (pigment), Boston University Red, Scarlet, Lust et Ferrari Red comme top 10. Toutes les distances du top 10 sont inférieures à 0.08.

Des recherches plus complexes peuvent également être réalisées en effectuant une jointure sur plusieurs critères. Ainsi, la recherche de lave vive (select c1.colorName from color c1 inner join color c2 on (c1.colorName=c2.colorName) where c1.searchedText like 'lava' and c2.searchedText like 'vivid') donne 57 résultats. La couleur dark terra cotta est ainsi écartée car bien que « proche » de lave (à une distance 0.09), elle est jugée comme non vive.

12.7 Conclusion

La problématique de recherche de couleurs dans une base de données est expliquée par un domaine de définition ouvert et soumis aux évolutions de la langue et aux spécificités des applications. Elle a pu être résolue d'une manière efficace par un calcul de relations entre les couleurs. Ces relations sont déterminées de manière automatique depuis des informations chromatiques disponibles librement sur le Web. Les relations sont calculées au moyen de modèles colorimétriques bien connus que sont TSL et TCL. Nous avons cependant vu les limites de ces modèles dans leur version standard et avons proposé une technique de transformation paramétrée afin d'y remédier. Nous avons enfin proposé de calculer des paramètres efficaces pour cette transformation au moyen d'un outil de test accessible en ligne. La disponibilité de cet outil permet notamment de contourner le problème de subjectivité des relations colorimétriques en donnant la possibilité au lecteur d'affiner les paramètres estimés idéaux.

Conclusion

Conclusion

Les objectifs de cette thèse étaient doubles : apporter des contributions scientifiques aux problèmes étudiés et développer des outils informatiques répondant aux besoins des clients d'Onyme et suffisamment fiables pour pouvoir être mis en production.

Nous mettons en évidence dans ce qui suit les principaux apports de la thèse, proposés pour répondre à ce double objectif.

Représentations de textes

Nous nous sommes intéressés d'abord aux représentations de textes courts d'opinions, des plus simples à construire comme les représentations par suites de caractères ou de mots, à des représentations complexes comme les représentations sémantiques. Nous avons explicité et justifié le choix de nous focaliser sur les représentations vectorielles de par le bon rapport qualité / temps de réalisation qu'elles offrent aussi bien en terme de temps CPU de construction de la représentation, qu'en terme de temps CPU de calcul du regroupement et de la classification. Cette partie de nos travaux a fait l'objet d'une publication à la conférence Rencontres des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL), édition 2010, Montréal, Canada (TROUVILLIEZ 2010). Des méthodes de simplification et de pondération de l'espace vectoriel ont été considérées afin de limiter la taille de ce dernier et de le rendre plus adéquat à nos traitements. Nos expériences ont montré l'intérêt d'employer une lemmatisation en combinaison avec une racinisation ou encore l'intérêt d'employer une pondération de l'espace vecto-

riel et une liste noire. Seul le traitement proposé de la négation comme portant sur l'intégralité du message nous a posé des problèmes. Ce constat est explicable par l'hypothèse faite que la majeure partie des verbatims ne comportent qu'une seule idée. Cette hypothèse n'est en réalité que peu vérifiée puisqu'entre 40% et 70% des verbatims comportent plusieurs idées selon les synthèses que nous souhaitons générer. Nous avons ainsi proposé d'enrichir la représentation vectorielle utilisée pour intégrer l'information que le message comporte ou non plusieurs idées à regrouper. Nous avons étudié la pertinence de critères statistiques comme la taille des messages ou le nombre de verbes dans le message dans cette reconnaissance. L'évolution des conditions industrielles d'application de nos algorithmes au cours de la thèse nous a conduit à nous intéresser aux questionnaires de type NPS comportant une note numérique comprise entre 0 et 10. Nous avons ainsi étudié diverses techniques de prise en compte de cette note dans la représentation des textes, de la représentation numérique de la note à une représentation alphabétique par classification en catégories NPS.

L'orthographe peu rigoureuse des messages que nous avons à traiter a conduit à proposer une correction orthographique basée sur des représentations des mots par suites de caractères et par suites de phonèmes. Cette approche a été validée notamment grâce au corpus WicoPaco, corpus de fautes d'orthographe construit à partir des révisions de Wikipédia.

Nous nous sommes enfin tournés vers la problématique de la transcription de recherches de couleurs exprimées en langage naturel en requêtes efficientes et compréhensibles par un système de gestion de base de données. Nous entendons par efficient que le résultat retourné par la requête doit correspondre à l'ensemble des résultats acceptables selon la recherche exprimée et non pas uniquement à une recherche stricte. Notre but fut d'établir des représentations de couleurs permettant des rapprochements par perception colorimétrique. Ces contraintes ont conduit à opter pour des représentations de type vectoriel.

Similarités

Afin d'effectuer des regroupements de textes, nous devions aborder les problématiques de calcul de relation de similarités entre eux. Le choix de représentations de type vectoriel est particulièrement intéressant à cet effet puisqu'il présente l'avantage d'offrir de telles relations de manière standard. Nous avons ainsi étudié entre autres les distances euclidienne et de Manhattan ainsi que la similarité du cosinus et avons proposé notre propre similarité textuelle. Nous nous sommes intéressés à l'apport des relations de similarités entre mots des textes basées sur des relations linguistiques telles que la synonymie, l'antonymie, l'hyperonymie. *Wordnet* et ses adaptations pour la langue française ont été exploités dans ce sens. Après l'étude des problèmes standards de similarités de représentations vectorielles de textes, deux problématiques propres à nos traitements ont été abordées. En premier lieu, nous nous sommes intéressés au regroupement et à la classification d'un même message selon plusieurs idées. Le but n'est pas de former une partition mais un recouvrement de l'espace d'entrée (regroupement et classification conceptuelle multi-classes non-mutuellement exclusives). Des adaptations d'algorithmes de regroupement et classification de la littérature pour gérer ces messages ont été proposées. En second lieu, nous avons étudié l'impact / l'apport d'une note NPS sur la similarité utilisée par la classification. Nous avons constaté que la note NPS permettait un gain sur la classification des questions 2 de questionnaires NPS.

L'utilisation de distances d'édition comme celle de Levenshtein appliquée à des représentations de mots par suites de caractères et par suites de phonèmes a permis de procéder à une correction orthographique des mots. Cette distance a été comparée à des distances plus strictes telle que la dissimilarité 0/1, utile pour des rapprochements par consonances.

Nous avons également abordé la problématique de la transcription de recherches de produits par des critères exprimés en langage naturel en requêtes compréhensibles par un système de gestion de base de données. Le premier objectif consistait en l'enrichissement de la connaissance spécifique du catalogue par des ressources généralistes. Nous avons explicité nos techniques de mise en relation du lexique contenu dans le catalogue avec le lexique généraliste d'une ressource *Wordnet*. Nous avons

étudié les cas de la langue française avec le WOLF et l'*EuroWordnet* FR ainsi que la langue anglaise avec l'*EuroWordnet* EN. Nous avons proposé une solution au manque de spécificité des ressources généralistes par rapport aux catalogues de produits en cherchant des relations d'hyponymie entre les deux lexiques. Cette partie de nos travaux a fait l'objet d'une publication à la conférence internationale Global Wordnet Conference (GWC), édition 2012, Matsue, Japon (TROUVILLIEZ 2012). Au delà de la mise en relation des lexiques, nous avons également abordé le problème de la transcription de requêtes complexes comportant des conjonctions et alternatives sur plusieurs critères. La gestion de ce type de requête en contexte orthographique peu rigoureux nous a conduit à étudier des modèles syntaxiques robustes aux erreurs d'orthographe.

En terme de recherche de couleurs, notre but fut d'établir des relations de similarité entre couleurs permettant des rapprochements par perception colorimétrique. La subjectivité des résultats dans un tel cadre rend leur validation à large échelle nécessaire. Cela nous a conduit à privilégier les distances intuitives pour un humain telles que les distances euclidienne ou de Manhattan. A partir de là, nous avons développé une procédure visant à construire une relation de proximité colorimétrique dans une base de données. Cette relation peut être enrichie facilement puisque construite à partir de ressources colorimétriques évolutives telle que Wikipédia. Cette partie de nos travaux a fait l'objet de la rédaction d'un article non encore publié.

Développement de programmes

Les travaux présentés dans cette thèse ont fait l'objet de la réalisation de 5 projets informatiques écrits en Java, dont 2 font l'objet aujourd'hui de traitements quotidiens en production. Ces 5 projets représentent au total près de 32 000 lignes de codes dont 23 000 lignes (72%) dans les deux projets principaux que sont le projet d'analyse des verbatims d'Onyme Opinions et le projet d'analyse des verbatims de l'agent conversationnel. Les deux projets faisant l'objet de traitements en production représentent 16 000 lignes (50%) de code environ. Le détail de ces chiffres est donné en annexe A page A-1.

Traitement des messages d'Onyme Opinions

L'outil développé pour le regroupement et la classification de textes d'opinions permet le traitement mensuel de près de 30 000 messages en classification. En moyenne, entre 10 et 15% de ces messages ne sont jamais vérifiés par un codificateur humain, soit entre 3 000 et 4 500 messages par mois. Ces messages sont donc traités entièrement par nos algorithmes. Au début de cette thèse, aucun algorithme de sélection des messages à vérifier n'était en place. La vérification devait alors se faire sur la totalité des messages, même les plus simples. Pour les autres messages, nos algorithmes proposent des idées qui peuvent être vérifiées ou non par un codificateur humain. Des mesures plus précises de la précision et du rappel sur les messages vérifiés humainement sont aussi envisagées. Par ailleurs, les perspectives d'évolution de l'offre Onyme Opinions laissent envisager une volumétrie de traitement mensuel de près de 60 000 messages en classification à l'horizon fin 2013. Dans cette optique, la sélection des messages à valider devient un enjeu encore plus important et la mise en place d'une heuristique de confiance est dans ce cadre envisagée. Le détail des chiffres est donné en annexe A page A-1.

Réalisation de prototypes

Les travaux réalisés dans le domaine de la recherche de produits et de couleurs dans une base de données à partir de requêtes en langage naturel ont donné lieu à la production de prototypes d'agent conversationnel vendeur et de calculateur de similarités colorimétriques. Le deuxième prototype est disponible en ligne : `http://labs.onyme.com/`.

Perspectives

Les perspectives à l'issue de cette thèse sont nombreuses tant sur le plan scientifique que sur le plan industriel. Nous en détaillons ici certaines.

Représentations de textes

Des améliorations peuvent être apportées aux représentations informatiques des textes utilisées dans ces travaux.

Une meilleure représentation des textes comportant plusieurs idées constitue un premier axe d'amélioration. L'approche proposée dans ce manuscrit se basant exclusivement sur la taille du message ne constitue pas une solution suffisamment robuste pour détecter correctement les textes comportant plusieurs idées. Pire, cette approche ne propose pas de solution pour nous aider à isoler les différentes idées les unes des autres lorsque cela est possible. Dans ce domaine, nous pensons qu'une analyse syntaxique de surface permettra la représentation isolée des idées lorsque celles-ci sont clairement séparées sur un plan syntaxique.

L'amélioration de la représentation des mots pour la correction orthographique constitue un second axe d'amélioration par :

1. l'utilisation de la co-occurrence de mots. Des modèles 3-grammes et 5-grammes peuvent être tentés. Il faut toutefois prendre garde à la taille de l'espace de représentation engendré ;
2. l'exploration de techniques relevant de la transcription langue écrite / langue parlée.

Similarités et désambiguïsation des mots

L'absence de désambiguïsation des mots dans la représentation des textes peut devenir problématique pour des mots très polysémiques tel que « être ». De manière générale, lorsqu'un mot apparaît dans une dizaine de *synsets* voir plus, notre stratégie de non-désambiguïsation pousse à considérer trop de rapprochements inopportuns. Il nous faut dans ce cas une stratégie de désambiguïsation. Une possibilité simple est de considérer un seuil d'acceptabilité de polysémie au delà duquel la recherche des *synsets* du mot échoue. Dans ce cas, le mot n'est similaire qu'à lui-même.

Concernant la recherche de produits dans une base de données à partir de requêtes en langage naturel, l'absence de désambiguïsation des mots peut poser des problèmes à la liaison des vocabulaires spécifiques et généralistes lorsque les termes du catalogue sont très polysémiques dans les ressources généralistes. Pour ces cas, beaucoup de liens entre les ressources sont créés dont des liens non pertinents. Cela, en plus d'alourdir le traitement par l'étude de proximité non intéressantes, peut engendrer la reconnaissance à tort d'un critère de recherche de produits. Par exemple, la requête « je cherche une souris pour mon mac » peut amener à proposer des rongeurs à l'internaute alors qu'une désambiguïsation du mot « souris » pourrait écarter cette hypothèse. En l'état, le prototype peut écarter les rongeurs uniquement s'il trouve une caractéristique correspondant à « mac » pour les souris d'ordinateur.

Sélection des verbatims à présenter au codificateur

Plus sur le plan industriel, des améliorations peuvent être apportées concernant la sélection des verbatims à présenter au codificateur par :

1. l'inclusion des verbatims complexes et comportant plusieurs idées, répétés de manière fréquente ;
2. l'établissement d'une relation de similarité permettant de décider si un verbatim doit être ou non présenté à un codificateur (similaire ou non à un verbatim connu). Des distances identiques à celles employées pour la correction orthographique peuvent être utilisées :

(a) distance d'édition de caractères ;

(b) distance d'édition phonétique.

Bibliographie et index

Livres

BLOOMFIELD L. (1933). *Language*. Holt (cf. p. 48).

CELKO J. (1995). *SQL avancé*. Thomson International Publishing (cf. p. 68).

COCKE J. (1969). *Programming languages and their compilers : Preliminary notes*. Courant Institute of Mathematical Sciences, New York University (cf. p. 51).

CORNUÉJOLS A. et MICLET L. (2010). *Apprentissage artificiel*. Eyrolles (cf. p. 101, 105, 109).

DE SAUSSURE F. (1916). *Cours de linguistique générale*. English translation : Course in General Linguistics. London : Peter Owen, 1960. Paris/Lausanne : v.C. Bally et A. Sechehaye (eds.) (cf. p. 5, 23).

FAIRON C., KLEIN J.R. et PAUMIER S. (2006). *Le langage SMS : étude d'un corpus informatisé à partir de l'enquête Faites don de vos SMS à la science*. Cental (cf. p. 70).

RASTIER F. (1991). *Sémantique et recherches cognitives*. Presses universitaires de France (cf. p. 258).

SALEM A. (1987). *Pratique des segments répétés : essai de statistique textuelle*. Klincksieck (cf. p. 28, 175).

SOWA J.F. (1984). *Conceptual structures*. Addison-Wesley Reading, MA (cf. p. 58).

VOSSEN P. (1998). *EuroWordNet : a multilingual database with lexical semantic networks for European languages*. Dordrecht : Kluwer (cf. p. 176, 263).

WEIL-BARAIS A. et DUBOIS D. (1994). *L'homme cognitif*. Collection Premier cycle. Presses universitaires de France. URL : http://books.google.fr/books?id=4Y4QAQAAIAAJ (cf. p. 89).

Actes

CANDITO M., NIVRE J. et al. (2010). « Benchmarking of statistical dependency parsers for French ». In : *Proceedings of the 23rd International Conference on Computational Linguistics : Posters*. Association for Computational Linguistics, p. 108–116 (cf. p. 57).

CANDITO M. et SEDDAH D. (juin 2012). « Le corpus Sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical (The Sequoia Corpus : Syntactic Annotation and Use for a Parser Lexical Domain Adaptation Method) [in French] ». In : *Actes de la conférence conjointe JEP-TALN-RECITAL 2012, volume 2 : TALN*. Grenoble, France : ATALA/AFCP, p. 321–334. URL : http://www.aclweb.org/anthology/F/F12/F12-2024 (cf. p. 56).

CARLSON Andrew et FETTE Ian (2007). « Memory-based context-sensitive spelling correction at web scale. » In : *ICMLA*. Sous la dir. de M. Arif WANI et al. IEEE Computer Society, p. 166–171 (cf. p. 43, 223, 224).

CRABBÉ B., CANDITO M. et al. (2008). « Expériences d'analyse syntaxique statistique du français ». In : *Traitement automatique des langues naturelles - TALN 2008*. Avignon, France (cf. p. 57).

ERTÖZ L., STEINBACH M. et KUMAR V. (2003). « Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data ». In : *SIAM international conference on data mining*, p. 47–58 (cf. p. 112).

HUNTER R. S. (juil. 1948). « Photoelectric Color-Difference Meter ». In : *Proceedings of the Winter Meeting of the Optical Society of America*. T. 38, p. 661 (cf. p. 92).

ISLAM Aminul et INKPEN Diana (2009). « Real-word spelling correction using google web 1tn-gram data set ». In : *In CIKM*, p. 1689–1692 (cf. p. 43, 223).

LEVENSHTEIN V.I. (1966). « Binary codes capable of correcting deletions, insertions, and reversals ». In : *Soviet physics doklady*. T. 10. 8, p. 707–710 (cf. p. 79).

LI Q., SHI Z. et LUO S. (2007). « Image Retrieval Based on Fuzzy Color Semantics ». In : *Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International*. IEEE, p. 1–5 (cf. p. 289).

LUND K., BURGESS C. et ATCHLEY R.A. (1995). « Semantic and associative priming in high-dimensional semantic space ». In : *Proceedings of the 17th annual conference of the Cognitive Science Society*. T. 17, p. 660–665 (cf. p. 43, 44).

MAX Aurélien et WISNIEWSKI Guillaume (2010). « Mining Naturally-occurring Corrections and Paraphrases from Wikipedia's Revision History ». Anglais. In : *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. Valletta, Malta : European Language Resources Association (ELRA). ISBN : 2-9517408-6-7 (cf. p. 219).

NIVRE J. (2003). « An efficient algorithm for projective dependency parsing ». In : *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*. Citeseer, p. 149–160 (cf. p. 57).

NIVRE J., HALL J. et NILSSON J. (2004). « Memory-based dependency parsing ». In : *Proceedings of CoNLL*, p. 49–56 (cf. p. 57).

PETROV S., BARRETT L. et al. (2006). « Learning accurate, compact, and interpretable tree annotation ». In : *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, p. 433–440 (cf. p. 57).

PETROV S. et KLEIN D. (2007). « Improved inference for unlexicalized parsing ». In : *Proceedings of NAACL HLT 2007*, p. 404–411 (cf. p. 57).

POIRIER D. et al. (2008). « Automating opinion analysis in film reviews : the case of statistic versus linguistic approach ». In : *Proceedings of the LREC 2008 Workshop on Sentiment Analysis : Emotion, Metaphor, Ontology and Terminology*, p. 94–101 (cf. p. 170, 180).

RILOFF E. (1995). « Little words can make a big difference for text classification ». In : *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM New York, NY, USA, p. 130–136 (cf. p. 25, 172).

SAGOT B. et FIŠER D. (2008). « Construction d'un wordnet libre du français à partir de ressources multilingues ». In : *Proceedings of TALN 2008* (cf. p. 176, 262, 263).

SALTON G. et BUCKLEY C (1988). « Term-weighting approaches in automatic text retrieval ». In : *INFORMATION PROCESSING AND MANAGEMENT*, p. 513–523 (cf. p. 29).

SCHMID H. (1994). « Probabilistic part-of-speech tagging using decision trees ». In : *Proceedings of International Conference on New Methods in Language Processing*. T. 12. Manchester, UK, p. 44–49 (cf. p. 34, 173, 266).

TROUVILLIEZ B. (2010). « Représentation vectorielle de textes courts d'opinions. Analyse de traitements sémantiques pour la fouille d'opinions par clustering ». In : *Actes de RECITAL 2010 (Rencontres des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues)*. ATALA. Montréal : RALI et POLYMTL (cf. p. 169, 315).

— (2012). « Linking specific and generalist knowledge. Building terminological ressources from sales catalogues and generalist resources ». In : *6th International Global Wordnet Conference (GWC'12)*, p. 357–364 (cf. p. 257, 318).

VERNIER Matthieu et al. (2007). « Classification de textes d'opinions : une approche mixte n-grammes et sémantique ». In : *Actes de l'atelier Défi Fouilles de Textes (DEFT'07)*. Grenoble, France, p. 93–108 (cf. p. 43).

WISNIEWSKI Guillaume, MAX Aurélien et YVON François (2010). « Recueil et Analyse d'un corpus écologique de corrections orthographiques extrait des révisions de Wikipédia ». In : *Actes de la 10eme conférence sur le Traitement Automatique des Langues Naturelles (TALN'2010)*. Montréal, Canada (cf. p. 219, 222–224, 262).

Articles

ABEILLÉ A., CLÉMENT L. et TOUSSENEL F. (2003). « Building a treebank for French ». In : *Treebanks*, p. 165–187 (cf. p. 49, 53, 56).

BERLIN Brent et KAY Paul (1969). « Basic Color Terms : their Universality and Evolution ». In : URL : `http://wals.info/refdb/record/6087` (cf. p. 295).

BOURIGAULT D., AUSSENAC-GILLES N. et CHARLET J. (2004). « Construction de ressources terminologiques ou ontologiques à partir de textes Un cadre unificateur pour trois études de cas. » In : *Revue d'Intelligence Artificielle* vol. 18.n° 1, p. 87–110 (cf. p. 258).

CHOMSKY N. (1956). « Three models for the description of language ». In : *IRE Transactions on Information Theory* vol. 2.n° 3, p. 113–124 (cf. p. 47).

DAMERAU F.J. (1964). « A technique for computer detection and correction of spelling errors ». In : *Communications of the ACM* vol. 7.n° 3, p. 171–176 (cf. p. 81).

DEERWESTER S. et al. (1990). « Indexing by latent semantic analysis ». In : *Journal of the American society for information science* vol. 41.n° 6, p. 391–407 (cf. p. 38).

EARLEY J. (1970). « An efficient context-free parsing algorithm ». In : *Communications of the ACM* vol. 13.n° 2, p. 94–102 (cf. p. 51).

GUILD J. (1932). « The colorimetric properties of the spectrum ». In : *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* vol. 230, p. 149–187 (cf. p. 89, 91).

HAMMING R.W. (1950). « Error detecting and error correcting codes ». In : *Bell System technical journal* vol. 29.n° 2, p. 147–160 (cf. p. 78).

JARO M.A. (1989). « Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida ». In : *Journal of the American Statistical Association*, p. 414–420 (cf. p. 83).

JOBLOVE George H. et GREENBERG Donald (1978). « Color spaces for computer graphics ». In : *SIGGRAPH Comput. Graph.* vol. 12.n° 3, p. 20–25. ISSN : 0097-8930. DOI : 10.1145/965139.807362. URL : http://doi.acm.org/10.1145/965139.807362 (cf. p. 95).

LLOYD S. (1982). « Least squares quantization in PCM ». In : *Information Theory, IEEE Transactions on* vol. 28.n° 2, p. 129–137 (cf. p. 136).

LUHN H.P. (1958). « The automatic creation of literature abstracts ». In : *IBM Journal of research and development* vol. 2.n° 2, p. 159–165 (cf. p. 29).

MARCUS M.P., MARCINKIEWICZ M.A. et SANTORINI B. (1993). « Building a large annotated corpus of English : The Penn Treebank ». In : *Computational linguistics* vol. 19.n° 2, p. 313–330 (cf. p. 56).

MILLER G.A. (1995). « WordNet : a lexical database for English ». In : *Communications of the ACM* vol. 38.n° 11, p. 41 (cf. p. 90, 176, 262).

PHILIPS L. (1990). « Hanging on the Metaphone ». In : *Computer Language* vol. 7 (cf. p. 71).

— (2000). « The double metaphone search algorithm ». In : *C/C++ Users Journal* (cf. p. 71).

PORTER M. (1980). « An Algorithm for Suffix Stripping ». In : *Program* vol. 14.n° 3, p. 130–137 (cf. p. 33).

RASTIER F. (1989). « Sens et textualité ». In : *Paris, Hachette* (cf. p. 58).

REICHHELD F.F. (2003). « The one number you need to grow ». In : *Harvard business review* vol. 81.n° 12, p. 46–55 (cf. p. 156, 226, 227).

SALTON G., WONG A. et YANG CS (1975). « A vector space model for automatic indexing ». In : *Communications of the ACM* vol. 18.n° 11, p. 613–620 (cf. p. 25, 28, 175).

SHANNON C.E. (1948). « A Mathematical Theory of Communication ». In : *Bell System Technical Journal* vol. 27, p. 379–423 (cf. p. 28, 41, 175).

SMITH T. et GUILD J. (1931). « The CIE colorimetric standards and their use ». In : *Transactions of the Optical Society* vol. 33, p. 73–135 (cf. p. 92).

SPARCK JONES K. (1972). « A statistical interpretation of term specificity and its application in retrieval ». In : *Journal of Documentation* vol. 28.n° 1, p. 11–20 (cf. p. 29, 175).

STANCHEV P., GREEN JR D. et DIMITROV B. (2003). « High level color similarity retrieval ». In : *International Journal « Information Theories and Applications »* vol. 10, p. 283–287 (cf. p. 288).

STOKES M. et al. (1996). « A standard default color space for the internet-srgb ». In : *Microsoft and Hewlett-Packard Joint Report* (cf. p. 92).

TESNIERE L. (1959). « Éléments de Syntaxe structurale ». In : *Klincksieck, Paris* (cf. p. 52).

WRIGHT W D (1929). « A re-determination of the trichromatic coefficients of the spectral colours ». In : *Transactions of the Optical Society* vol. 30.n° 4, p. 141–164. URL : `http://stacks.iop.org/1475-4878/30/i=4/a=301` (cf. p. 89, 91).

YOUNG T. (1802). « The Bakerian Lecture : On the theory of light and colours ». In : *Philosophical transactions of the Royal Society of London* vol. 92, p. 12–48 (cf. p. 89, 91).

YOUNGER D.H. (1967). « Recognition and parsing of context-free languages in time n3* ». In : *Information and control* vol. 10.n° 2, p. 189–208 (cf. p. 51).

Sites Internet

BROUARD F. (2004). *L'art des « Soundex »*. URL : http://sqlpro.developpez.com/cours/soundex/ (cf. p. 68, 72).

FRANZ Alex et BRANTS Thorsten (2006). *All Our N-gram are Belong to You*. URL : http://googleresearch.blogspot.fr/2006/08/all-our-n-gram-are-belong-to-you.html (cf. p. 43).

Autres sources

KASAMI T. (1965). *An efficient recognition and syntaxanalysis algorithm for context-free languages*. Rapp. tech. DTIC Document (cf. p. 51).

ODELL Margaret K. et RUSSELL Robert C. (1918/1922). « U.S. Patents 1261167 (1918), 1435663 (1922) ». Cited in Knuth (1973) (cf. p. 66).

WINKLER W.E. (1999). *The state of record linkage and current research problems*. Rapp. tech. Statistical Research Division, U.S. Census Bureau (cf. p. 84).

Annexes

Annexe A

Métriques clés des projets

Ce chapitre présente quelques-unes des métriques clés des principaux projets informatiques ayant faisant l'objet d'une étude dans cette thèse. Ces mesures ont été réalisées à l'aide de l'outil libre d'analyse de code Sonar [1]. Nous faisons par ailleurs ici la distinction entre les projets faisant l'objet de traitements en production et ceux ayant fait l'objet de la réalisation d'un prototype en collaboration ou non avec une société tiers.

A.1 Métriques clés des projets faisant l'objet de traitements en production

A.1.1 Métriques clés du projet d'analyse des verbatims d'Onyme Opinions

Le projet d'analyse des verbatims d'Onyme Opinions a été développé dans le cadre du logiciel Onyme Opinions. Il est utilisé quotidiennement par la société Onyme SARL pour le traitement des commentaires clients. Le tableau suivant donne sur chaque ligne les volumes de verbatims traités par le logiciel par an ou par mois et ce, en 2012 ou en estimation pour 2013.

1. http://www.sonarsource.org/

2012	**# verbatims traités par an**	315 120
	# verbatims traités par mois en moyenne	26 260
Estimation pour 2013	**# verbatims estimés traités par an**	695 620
	# verbatims estimés traités par mois en moyenne	57 968

Le tableau suivant donne sur chaque ligne une statistique relevée par Sonar sur le projet d'analyse des verbatims d'Onyme Opinions.

# lignes dans le programme	24 154
# lignes de code	12 908
# instructions	6 062
# fichiers	183
# classes	187
# packages	30
# méthodes	1 274

Nous concluons avec ces métriques en donnant quelques précisions quand aux temps de traitement CPU de l'algorithme par proche voisinage, algorithme utilisé actuellement pour les traitements quotidiens du logiciel Onyme Opinions.

Question	**Nombre de verbatims à traiter**	**Temps de classification**	**Temps de traitement global**	**Temps de classification moyen par verbatim à traiter**	**Temps de traitement moyen par verbatim à traiter**
2 - Pourquoi cette note ?	2900	17 min	18 min	0.35 s/v	0.37 s/v
3 - Que peut-on améliorer ?	2240	23 min	24 min	0.62 s/v	0.64 s/v
2 - Pourquoi cette note ?	139	2 min	3 min	0.86 s/v	1.29 s/v
3 - Que peut-on améliorer ?	147	45 s	1 min	0.31 s/v	0.41 s/v

A.1.2 Métriques clés du projet de correction orthographique

Le projet de correction orthographique a été développé principalement dans le cadre du logiciel Onyme Opinions. Il est utilisé quotidiennement par la société Onyme pour le traitement des commentaires clients. Le tableau suivant donne sur chaque ligne une statistique relevée par Sonar sur le projet de correction orthographique.

# lignes dans le programme	4 419
# lignes de code	2 967
# instructions	1 506
# fichiers	35
# classes	35
# packages	9
# méthodes	132

A.2 Métriques clés des projets ayant fait l'objet de la réalisation d'un prototype

A.2.1 Métriques clés du projet d'analyse des verbatims de l'agent conversationnel

Le projet d'analyse des verbatims de l'agent conversationnel a fait l'objet de la réalisation d'une implémentation dans le cadre d'une prestation avec une société tiers. A notre connaissance, le prototype réalisé ne fait actuellement pas l'objet de traitements en production. Le tableau suivant donne sur chaque ligne une statistique relevée par Sonar sur le projet d'analyse des verbatims de l'agent conversationnel.

# lignes dans le programme	16 415
# lignes de code	9 823
# instructions	4 903
# fichiers	112
# classes	118
# packages	22
# méthodes	653

A.2.2 Métriques clés du projet de recherche de couleurs dans une base de données

Le projet de recherche de couleurs dans une base de données ne fait pas l'objet de traitements en production. Le prototype est en revanche disponible en ligne sur le labs de la société Onyme SARL[2] à des fins de test. Le tableau suivant donne sur chaque ligne une statistique relevée par Sonar sur le projet de recherche de couleurs dans une base de données.

# lignes dans le programme	3 738
# lignes de code	2 413
# instructions	1 303
# fichiers	27
# classes	28
# packages	5
# méthodes	230

A.2.3 Métriques clés du projet d'exploration de ressources linguistiques « *Wordnet* »

Le projet d'exploration de ressources linguistiques « Wordnet » a été développé dans le but de parcourir visuellement les liens d'hyper/hyponymie dans une ressource de type « *Wordnet* » telle que le WOLF ou nos ressources spécifiques construites à partir de catalogues de produits. Le tableau suivant donne sur chaque ligne une

2. `http://labs.onyme.com/`

statistique relevée par Sonar sur le projet d'exploration de ressources linguistiques « *Wordnet* ».

# lignes dans le programme	5 951
# lignes de code	3 842
# instructions	1 948
# fichiers	32
# classes	32
# packages	11
# méthodes	274

Annexe B

Exemples

B.1 Extraits de l'ensemble d'apprentissage supervisé

Exemple B.1

Extrait de l'ensemble d'apprentissage supervisé pour l'optimisation et l'évaluation des paramètres textuel et alphanumérique

Prix abordable

Les prix

Prix

de bon prix

prix abordables

Prix attractifs

prix intéressants

les prix accessibles

les prix raisonnables

vêtements à bons prix

les prix sont corrects

les prix sont plus bas.

prix sont raisonnable

Large choix

beaucoup de choix

choix

le choix des produits

Il y a du choix

choix des articles

Bon choix d'articles

LE CHOIX !

Bon rapport qualité/prix

Bon rapport qualité prix

bon rapport qualité prix

Rapport qualité prix

le rapport qualité prix

Bonne organisation / agencement du magasin

Bien agence et bien ranger

le magasin est bien ranger

Le magasin est bien adjensé

Bon agencement de la boutique

vetements bien classés par âge

il est agencé, on se repere vite

la bonne organisation des rayons

c'est l'ajansement de la boutique

Magasin bien consut et pratique !

Tout est bien rangé, bien présenter

Bon accueil / accueil souriant

bon ACCUEIL

Très bon accueil

accueil

l'accueil

Manque de choix

un choix vestimentaire limité

Pas trop de choix en boutique

manque de choix de chaussures

Peu de choix bébé / catalogue

Pas assez de référence en magasin

Assez peu de choix pour les bébés.

peu de choix

pas assez de choix

Qualité insatisfaisante

LA qualité des produit

qualité moyenne

Qualité du tissus

La qualité inégale

La qualité des basics

la qualité de vos vêtements

Mauvaise qualité des meubles

Qualité quelquefois médiocre.

la qualité des textiles n'est pas très bonne

J'ai été déçu par la qualité des articles achetés

car parfois la qualite des vetement sont pas terrible

B.2 Extraits des ensembles de test pour l'apprentissage supervisé

Exemple B.2

Extrait de l'ensemble de test d'apprentissage pour l'optimisation des paramètres textuel et alphanumérique et l'optimisation du rayon de voisinage

articles de qualité à prix accessibes

choix et prix

les prix

Prix

Choix varie et prix abordable

Grand magasin, aéré, avec un vaste choix, ce qui est agréable. En revanche, quel que soit le jour ou l'heure, on fait la queue, on passe beaucoup trop de temps à la caisse : les caissières ne sont pas du tout rapides et seules 2 caisses sont ouvertes.

grand magasin, du choix

J'y trouve très souvent ce que je cherche et même quand je ne cherche rien de particulier.

l'accueil et le choix

l'accueil, le choix, l'espace ....

Large choix Bon accueil

Le choix des articles est attrayant

le personnel est très gentil, souriant. le choix est satisfaisant.

le rapport qualité prix des produits,la variété des produits proposés,l'accueil et les conseils des membres du magasin

un choix assez varié

Accueil, choix

ACCUEIL CHOIX

choix

choix correct.

choix,prix

Choix, qualité

le choix

LE CHOIX !

Le choix varie

Collections très sympas, problème de disponibilité de certains produits

j'aime les coloris et les coupes

j'aime vos produits, de plus il y a souvent des offres promo intéressantes

L'acceuil et les produit auraient mérité un9 mais la chaleur dans la boutique est insoutenable

la forme de l habit, la couleur, tout quoi !

La jeune fille qui m'a encaissé n'était pas aimable du tout ne faisait aucun effort de compréhension et me faisait répéter, je n'ai pourtant pas de problème de prononciation !!!! vaut mieux faire ces achats en ligne au moins on ne rencontre pas se genre de problème. Sinon je suis ravie de mon article choisi. bien avec les primotion.

le conseil très bon de la vendeuse et de bons produits pour les enfants

Les nouvelles tailles ajustables sont vraiment géniale, car quand on a une petite fille très mince c'est souvent compliqué de trouver !!

les vêtements sont sympas et de bonne qualité ... cependant un peu chers !!!!

Qualité et bon goût des produits

Très bon produits originaux et surtout de très bonne qualitee

Vêtements très sympas Souvent de jolis ensemble déjà coordonnés

beau désigne

beaux produits

bonne tenue

Bon produit

joli designe

jolie vetement

les articles

les produits

l'esthétique

bon rapport qualité prix

Localisation Prix qualité

Produits sympas, et bon rapport qualite / prix

un bon rapport qualité prix des produits un grand choix de modèles, il y en a pour tous les goûts

VÊTEMENTS SYMPAS POUR MES FILLES, BON RAPPORT QUALITÉ PRIX ET TOUJOURS UN SUPER ACCUEIL AU MAGASIN DE NANCY AVEC UNE ÉQUIPE TRÈS DISPONIBLE POUR LE CONSEIL ET TRÈS SOURIANTE.

articles de qualité avec remise

Bonne qualité des vetements. Prix tres abordables

conseil de la part de la vendeuse , accueil chaleureux et produit de très bonne qualité

Depuis toujours bonne qualité des matières des produits et délai rapide pour faire venir en magasin un article non disponible dans la taille désirée cette semaine.

je trouve qu'il y a beaucoup de choix, c'et agréable et la qualité est satisfaisante

La proximité du magasin par rapport à mon lieu d'habitation La qualité des produits L'environnement agréable du magasin La disponibilité et l'amabilité des vendeuses

la qualité et la diversité des vêtement.

les prix et la qualité

Prix choix qualité

Produits sympas, pratiques et qui tiennent bien. Bon rapport qualité/prix

Qualité des produits

Qualité des produits et du service

tout l 'équipement et le matériel de la chambre pour enfant ,et la qualité des vetements.

vêtement de qualité, jamais déçue...

bonne qualité

la qualite

Exemple B.3

Extraits du petit ensemble de test d'apprentissage pour l'évaluation des paramètres textuel et alphanumérique et l'évaluation du rayon de voisinage

Il va bien à mon fils

bonne présentation, soldes bien rangées, classées par taille, même en fin de journée!!

vendeuse très agréable et très souriante, à disposition

Bonne qualité et vêtements sympas

bon choix dans les vêtements, bon choix dans la puériculture

le prix le choix

La qualité, le choix et le prix

Je me suis fait avoir car la remise de 20% qui m'avait été proposée par mail n'a pas été appliquée alors que je suis bien venu le premier jour des soldes, mais il aurait fallu que j'imprime le mail!!! (bonjour l'image ecologique). Conclusion, j'ai rendu un des artciles acheté, et si je n'ai pas de compensation sur ce problème, je ne reviendrai plus.

La qualité des produits, et le fait que dans cette boutique il y a maintenant une petite cabane où mes enfants ont pu jouer pendant que je faisais mes achats, ils n'ont pas vu le temps passer et ne voulaient même plus partir et moi j'ai été tranquille pour tout regarder.

La qualité et l'esthétique des produits

La qualité et la beauté de la collection

amabilité, renseignements donnés

qualité de produits

proximité

qualité et style des produits

la gentillesse des vendeuses la qualité des vêtements (assez résistant) et taille très bien

TARIFS QUALITE DES VETEMENTS PERSONNEL ACCUEILLANT

Les vêtements sont très jolis et originaux, il y beaucoup de choix.

+ Personnel très aimable avec les parents et les enfants. - Petit magasin, même si c'est bien rangé, on se sent à l'étroit et il manque tout ce qui est déco de la chambre

Je n'ai pas trouver de jeans pour un enfant de 18 mois.

les vetement que j ai trouves

Amabilité du personnel, disponible.

la qualité, le prix

le + : la qualité des vêtements

la qualité des vendeuses

la qualité des articles

J ai demander en caisse qu on déduise mon chèque fidélité , on m à répondu oui pas de problème . Puis la personne à oublié et ma répondu : désolée je N avais pas compris , euh j ai oublié ... Demandez le la prochaine fois "!!!"

LA QUALITE DES VETEMENTS

j'aime les artcles que propose le magasin

rapport qualite/prix et vetements mignons

Des produits au design intéressant et de bonne qualité

magasin bien disposé

L'accueil et le service sont très bien.Il y a d'ailleurs une nouvelle vendeuse qui est très chaleureuse et competente !

J'ai des enfants pas facile à habiller un un peu trop costaut et l'autre un peu trop maigre et il n'y a que chez vous ou je trouve des pantalons tailler pour tous .

1/ la qualité 2/ les tailles intermédiaires 3/ la présentation des collections et leurs assortiments 4/ la qualité des chaussures en cuire à des prix très corrects

articles de bonne qualité

Bon accueil

accueil et produits

Accueil du personnel, bon rapport qualité prix

magasin mal agencé et personnel débordé

les prix.

bonne qualité des produits, accueil très appréciable ,

Exemple B.4

Extraits du grand ensemble de test d'apprentissage pour l'évaluation des paramètres textuel et alphanumérique et l'évaluation du rayon de voisinage

Le choix, le prix Un bémol pour les couleurs que je ne trouve pas tjs très jolie

les produits me plaisent, les prix après réductions sont attrayants

Bon conseil en magasin et bons produits à des prix raisonnables

qualité - choix des articles

produit de bonne qualité et prix raisonnable..

beaucoup de vendeurs, mais pourtant il est très difficile d'avoir des infos + le temps d'attente aux caisses est très fréquemment long ! conclusion : je repose généralement ce que j'ai choisi pour finalement le commander sur internet !

qualite de l accueil et du conseil

Qualité

Bon accueil. Vendeuse à l'écoute et très patiente

je ne suis pas totalement satisfaite de la qualité de certains produits, ni du style . Surtout les textiles filles (pulls, gilets) . les produits garçons me semblent plus convaincants.

Beauté des produits proposés, ensembles bien assortis, magnifiques chambres de bébés possibles. Prix promotionnels attractifs.

Mieux que sur le site

Accueil très sympathique, prix abordables

Moins mauvais acceuil que d habitude, longue attente en caisses

j'étais venue acheter des pyjamas avec un motif de Noël, je les ai tout de suite trouvés à l'entrée du magasin, exposés sur un présentoir spécial, avec les tailles recherchées et de jolis motifs.

très bon accueil et jolis articles à prix abordable

ACCUEIL SYMPATHIQUE

qualite des vetements / service et accueil en magasin

nombreux choix

Bonne qualité, plusieurs produits (linge, vetement, chaussure..etc)

Je trouve dommage que les réductions proposées sur internet ne soient pas aplliquées en magasin.

Tres bon accueil .vendeuse disponible et souriante.

Le modèle ne correspond pas à celui présenté sur catalogue

bien acceuillie bien guidée service rapide et efficace en caisse (demande de la carte)

la qualité des produits et leur originalité, la qualité d'accueil et de communication des vendeuses.

Bonne qualité des produits pour un prix non excessif

j'ai trouvé l'article que je souhaitais dans la taille que je souhaitais

la qualité des vêtements , et l'acceuil des vendeuse du magasin d'ancenis .

j'ai trouvé qu'il n'y avait pas beaucoup de choix ; je cherchais notamment des gilets pour différents âges fille et garçon...

l'accueil et la proprete et clarte du magasin

Les promotions toute au long de l'année sont intéressantes ainsi que le cumul des points grâce à la carte fidelité donnant droit à 10€ de réduction. J'apprécie de pouvoir réserver des articles et de ne pas payer de frais de port si je commande en magasin. Je ne trouve nul part ailleurs d'aussi beau tour de lit et gigoteuses ! Seul bémol : les prix sont assez élevés sur certains produits. Le sav est aussi très bien.

magasin agreable,vendeuse disponible.

Qualité des textiles, choix original de vêtements, très beaux linges de lit, d éco pour chambre enfant très sympa !

le prix, le choix

bon rapport qualité prix, proximité domicile

je n'ai jamais été déçue

Prix

Annexe C

Algorithmes

C.1 Algorithmes phonétiques

C.1.1 Soundex

L'algorithme Soundex procède selon six étapes décrites ci-dessous depuis le mot d'origine pour trouver sa représentation Soundex :

1. Représentation consonantique
   (a) Retranscrire le mot en majuscules
   (b) Conserver la première lettre du mot
   (c) Éliminer toutes les voyelles, le H et le W
2. Encodage alphanumérique
   (a) Transcoder les lettres restantes à l'aide de la table de correspondance de la langue
   (b) Éliminer toutes les paires consécutives de chiffres dupliquées
   (c) Conserver les quatre premiers caractères du Soundex ainsi obtenu et le compléter par des zéros le cas échéant

Le tableau C.1 donne les correspondances entre les lettres et le code Soundex pour les langues anglaise et française.

Lettre	**Type de consonance**	**Code Soundex**
Version langue anglaise		
B F P V	Bilabiales	1
C G J K Q S X Z	Labiodentales	2
D T	Dentales	3
L	Alvéolaires	4
M N	Vélaires	5
R	Laryngales	6
Version langue française		
B P	Bilabiales	1
C K Q	Labiodentales	2
D T	Dentales	3
L	Alvéolaires	4
M N	Vélaires	5
R	Laryngales	6
G J	Labiodentales	7
S X Z	Labiodentales	8
F V	Bilabiales	9

TABLE C.1 – Correspondances entre les lettres et le code Soundex pour les langues anglaises et françaises. L'ensemble des consonnes données sur chaque ligne de la 1re colonne est remplacé par le code Soundex correspondant donné dans la 3e colonne. Les correspondances pour les langues anglaise et française sont données dans chacune des deux parties du tableau.

C.1.2 Soundex 2

La version pour le français de Soundex 2 se compose des 14 étapes énumérées ci-dessous :

1. Représentation consonantique
   (a) Éliminer les blancs de part et d'autre
   (b) Convertir le mot en majuscule
   (c) Convertir les lettres accentuées et le ç en lettres non accentuées
   (d) Éliminer les blancs et les tirets
   (e) Remplacer les groupes de lettres par leur correspondance selon le tableau C.2 (en conservant l'ordre du tableau)
   (f) Remplacer les voyelles E, I, O et U par A exceptée pour le premier caractère conservé tel que.
   (g) Remplacer les groupes de lettres en préfixe par leur correspondance selon le tableau C.3
   (h) Remplacer les groupes de lettres par leur correspondance selon le tableau C.4
   (i) Supprimer les H sauf s'ils sont précédés de C ou S
   (j) Supprimer les Y sauf s'ils sont précédés d'un A
   (k) Supprimer les terminaisons suivantes A, T, D et S
   (l) Enlever tous les A sauf le A de tête s'il y en a un
2. Encodage alphanumérique
   (a) Enlever toutes les sous-chaînes de lettres répétitives
   (b) Conserver les 4 premiers caractères du mot et si besoin le compléter avec des zéros pour obtenir 4 caractères

Groupe de lettres	GUI	GUE	GA	GO	GU	CA	CO	CU	Q	CC	CK
Correspondant Soundex 2	KI	KE	KA	KO	K	KA	KO	KU	K	K	K

TABLE C.2 – Correspondances de groupes de lettres dans la méthode Soundex 2. Chaque colonne donne une correspondance entre le groupe de lettres à remplacer en 1re ligne avec le code Soundex 2 en 2e ligne.

Groupe de lettres	MAC	ASA	KN	PF	SCH	PH
Correspondant Soundex 2	MCC	AZA	NN	FF	SSS	FF

TABLE C.3 – Correspondances de groupes de lettres en préfixe dans la méthode Soundex 2. Chaque colonne donne une correspondance entre le groupe de lettres à remplacer en 1re ligne avec le code Soundex 2 en 2e ligne.

C.1.3 Phonex

L'algorithme Phonex se compose des 18 étapes suivantes :

1. Représentation par suite finie de phonèmes
   (a) Remplacer les y par des i
   (b) Supprimer les h qui ne sont pas précédés de c, s ou p
   (c) Remplacer les ph par f
   (d) Remplacer les groupes de lettres suivantes

Groupe de lettres	gan	gam	gain	gaim
Correspondant Phonex	kan	kam	kain	kaim

   (e) Remplacer les occurrences suivantes, si elles sont suivies par une lettre a, e, i, o ou u :

Groupe de lettres	ain	ein	aim	eim
Correspondant Phonex	yn	yn	yn	yn

Groupe de lettres	ASA	KN	PF	SCH	PH
Correspondant Soundex 2	AZA	NN	FF	SSS	FF

TABLE C.4 – Correspondances de groupes de lettres dans la méthode Soundex 2. Chaque colonne donne une correspondance entre le groupe de lettres à remplacer en 1re ligne avec le code Soundex 2 en 2e ligne.

(f) Remplacer les groupes de 3 lettres suivants (sons 'o', 'oua', 'ein') :

Groupe de lettres	eau	oua	ein	ain	eim	aim
Correspondant Phonex	o	2	4	4	4	4

(g) Remplacer le son 'é'

Groupe de lettres	é	è	ê	ai	ei	er	ess	et
Correspondant Phonex	y	y	y	y	y	yr	yss	yt

(h) Remplacer les groupes de 2 lettres suivantes (son 'an' et 'in'), sauf s'ils sont suivis par une lettre a, e, i o, u ou un son 1 à 4 :

Groupe de lettres	an	am	en	em	in
Correspondant Phonex	1	1	1	1	4

(i) Remplacer les s par des z s'ils sont suivis et précédés des lettres a, e, i, o, u ou d'un son 1 à 4

(j) Remplacer les groupes de 2 lettres suivants

Groupe de lettres	oe	eu	au	oi	oy	ou
Correspondant Phonex	e	e	o	2	2	3

(k) Remplacer les groupes de lettres suivants

Groupe de lettres	ch	sch	sh	ss	sc
Correspondant Phonex	5	5	5	s	s

(l) Remplacer le c par un s s'il est suivi d'un e ou d'un i

(m) Remplacer les lettres ou groupe de lettres suivants

Groupe de lettres	c	q	qu	gu	ga	go	gy
Correspondant Phonex	k	k	k	k	ka	ko	ky

2. Encodage alphanumérique de la suite de phonèmes

(a) Remplacer les lettres suivantes

Lettre	a	d	p	j	b	v	m
Correspondant Phonex	o	t	t	g	f	f	n

(b) Supprimer les lettres dupliquées

(c) Supprimer les terminaisons t et x

(d) Affecter à chaque lettres le code numérique correspondant

Lettre ou chiffre	1	2	3	4	5	e	f	g	h	i	k
Correspondant numérique Phonex	0	1	2	3	4	5	6	7	8	9	10

Lettre ou chiffre	l	n	o	r	s	t	u	w	x	y	z
Correspondant numérique Phonex	11	12	13	14	15	16	17	18	19	20	21

(e) Convertir les codes numériques ainsi obtenu en un nombre de base 22 exprimé en virgule flottante.

C.2 Algorithmes de regroupement et classification par apprentissage

Input : S : échantillon d'apprentissage, K : entier
Output : W : ensemble des classes

```
1  begin
2      W ← {};
3      M ← {};
4      for i from 1 to K do
5          w_i ← {};
6          W ← W ∪ w_i;
7          M ← M ∪ (w_i, initPseudoRandomMean());
8      end
9      continue ← true;
10     while continue do
11         foreach s in S do
12             bestM ← nil;
13             foreach m in M do
14                 if bestM = nil or dist(s, m.μ) < dist(s, bestM.μ) then
15                     bestM ← m;
16                 end
17             end
18             bestM.w ← bestM.w ∪ s;
19         end
20         continue ← false;
21         foreach m in M do
22             oldμ ← m.μ;
23             m.μ ← computeMean(m.w);
24             if oldμ != m.μ then
25                 continue ← true;
26             end
27         end
28     end
29     return W;
30 end
```

Algorithme 10 : Algorithme de regroupement par apprentissage des K-moyennes (*K-means*)

Input : S : échantillon d'apprentissage, seuil : réel
Output : W : ensemble des classes

```
1  begin
2      W ← initSingletonClustersForEachData(S);
3      continue ← true;
4      while continue do
5          bestcouple ← (nil, nil);
6          foreach w1 in W do
7              foreach w2 in W do
8                  if bestcouple = (nil,nil) or dist(w1, w2) < dist(bestcouple[1],
                   bestcouple[2]) then
9                      bestcouple ← (w1, w2);
10                 end
11             end
12         end
13         if dist(bestcouple[1], bestcouple[2]) < seuil then
14             W ← (w1 ∪ w2) ∪ W \ {w1, w2};
15             if |clusters| = 1 then
16                 continue ← false;
17             end
18         end
19         continue ← false;
20     end
21     return W;
22 end
```

Algorithme 11 : Algorithme de regroupement par apprentissage hiérarchique ascendant

Input : S : échantillon d'apprentissage, seuil : réel
Output : W : ensemble des classes

```
1  begin
2      W ← initClusterForData(S);
3      continue ← true;
4      while continue do
5          lesscoherent ← nil;
6          foreach w in W do
7              if lesscoherent = nil or coherence(w) < coherence(lesscoherent) then
8                  lesscoherent ← w;
9              end
10         end
11         if coherence(lesscoherent) < seuil then
12             W ← scission(lesscoherent) ∪ W \ {lesscoherent};
13             if |clusters| = |data| then
14                 continue ← false;
15             end
16         else
17             continue ← false;
18         end
19     end
20     return W;
21 end
```

Algorithme 12 : Algorithme de regroupement par apprentissage hiérarchique descendant

Input : S : échantillon d'apprentissage, x : donnée à classer

Output : w : une classe

```
1 begin
2     foreach (x_i, w_i) ∈ S do
3         D(x_i, x) ← compteDistance(x_i, x);
4     end
5     foreach K plus petites distances entre x_i et x do
6         Compter le nombre d'occurrence de chaque classe w_i;
7     end
8     return La classe w_i qui apparaît le plus souvent;
9 end
```

Algorithme 13 : Algorithme de classification par apprentissage des K-plus proches voisins

Résumé

Cette thèse porte sur l'établissement de similarités de données textuelles dans le domaine de la gestion de la relation client. Elle se décline en deux parties :

– l'analyse automatique de messages courts en réponse à des questionnaires de satisfaction ;

– la recherche de produits à partir de l'énonciation de critères au sein d'une conversation écrite mettant en jeu un humain et un programme agent.

La première partie a pour objectif la production d'informations statistiques structurées extraites des réponses aux questions. Les idées exprimées dans les réponses sont identifiées, organisées selon une taxonomie et quantifiées.

La seconde partie vise à transcrire les critères de recherche de produits en requêtes compréhensibles par un système de gestion de bases de données. Les critères étudiés vont de critères relativement simples comme la matière du produit jusqu'à des critères plus complexes comme le prix ou la couleur.

Les deux parties se rejoignent sur la problématique d'établissement de similarités entre données textuelles par des techniques de TAL. Les principales difficultés à surmonter sont liées aux caractéristiques des textes, rédigés en langage naturel, courts, et comportant fréquemment des fautes d'orthographe ou des négations. L'établissement de similarités sémantiques entre mots (synonymie, antonymie, etc) et l'établissement de relations syntaxiques entre syntagmes (conjonction, opposition, etc) sont également des problématiques abordées. Nous étudions également dans cette thèse des méthodes de regroupements et de classification automatique de textes afin d'analyser les réponses aux questionnaires de satisfaction.