université PARIS-SACLAY

NNT : 2019SACLX051

Thèse de doctorat

Toward Automatic Fact-Checking of Statistic Claims

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'École Polytechnique

École doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Palaiseau, le 26.09.2019, par

TIEN-DUC CAO

Composition du Jury :

Philippe Pucheral Professeur, UVSQ et Inria	Président du jury
Nathalie Aussenac-Gilles Directrice de recherche, IRIT	Rapporteur
Paolo Papotti Maître de Conférences, EURECOM	Rapporteur
Julien Leblay Chercheur, AIST, Japon	Examinateur
Philippe Lamarre Professeur, ENSA Lyon	Examinateur
Ioana Manolescu Directrice de recherche, Inria et Ecole Polytechnique	Directeur de thèse
Xavier Tannier Professeur, Paris Sorbonne Universités	Co-directeur de thèse

Contents

1 Introduction **7**
- 1.1 Motivation . . . 7
- 1.2 Contributions and outline . . . 8

2 Preliminaries **11**
- 2.1 Resource Description Framework . . . 11
- 2.2 Information extraction . . . 12
  - 2.2.1 Information extraction tasks . . . 12
  - 2.2.2 Machine learning for information extraction . . . 13
  - 2.2.3 Deep learning for information extraction . . . 14
  - 2.2.4 Metrics for evaluating information extraction quality . . . 15
  - 2.2.5 Text representation . . . 17
- 2.3 Conclusion . . . 18

3 State of the art of computational fact checking **19**
- 3.1 Claim extraction . . . 21
  - 3.1.1 Unsupervised approaches . . . 21
  - 3.1.2 Supervised methods . . . 22
- 3.2 Reference source search . . . 26
- 3.3 Related datasets . . . 26
- 3.4 Claim accuracy assessment . . . 27
  - 3.4.1 Using external sources . . . 27
  - 3.4.2 Using a knowledge graph . . . 30
  - 3.4.3 Using linguistic features . . . 32
  - 3.4.4 Using user input . . . 34
- 3.5 Fact checking challenges . . . 37
  - 3.5.1 Fake news challenge . . . 37
  - 3.5.2 Fact Extraction and VERification . . . 37
  - 3.5.3 Check worthiness . . . 38
- 3.6 Automated end-to-end fact checking systems . . . 38
- 3.7 Conclusion . . . 40

4 Extracting linked data from statistic spreadsheets **43**
- 4.1 Introduction . . . 43
- 4.2 Reference statistic data . . . 44
  - 4.2.1 INSEE data sources . . . 44
  - 4.2.2 Conceptual data model . . . 46
- 4.3 Spreadsheet data extraction . . . 48

4.3.1 Data cell identification . . . 48
4.3.1.1 The leftmost data location . . . 48
4.3.1.2 Row signature . . . 49
4.3.1.3 Collect additional data cells . . . 49
4.3.2 Identification and extraction of header cells . . . 49
4.3.2.1 The horizontal border . . . 49
4.3.2.2 Cell borders . . . 50
4.3.2.3 Collect header cells . . . 50
4.3.3 Populating the data model . . . 50
4.4 Linked data vocabulary . . . 51
4.5 Evaluation . . . 51
4.6 Implementation . . . 52
4.7 Related works . . . 52
4.8 Conclusion and future works . . . 53

5 Searching for truth in a database of statistics 55
5.1 Introduction . . . 55
5.2 Search problem and algorithm . . . 56
5.2.1 Dataset search . . . 57
5.2.2 Text processing . . . 57
5.2.3 Word-dataset score . . . 57
5.2.4 Relevance score function . . . 58
5.2.4.1 Content-based relevance score function . . . 59
5.2.4.2 Location-aware score components . . . 59
5.2.4.3 Content- and location-aware relevance score . . . 60
5.2.5 Data cell search . . . 60
5.3 Evaluation . . . 61
5.3.1 Datasets and queries . . . 61
5.3.2 Experiments . . . 62
5.3.2.1 Evaluation metric . . . 62
5.3.2.2 Parameter estimation and results . . . 62
5.3.2.3 Running time . . . 63
5.3.2.4 Comparison against baselines . . . 64
5.3.3 Web application for online statistic search . . . 65
5.4 Implementation . . . 65
5.5 Related works . . . 66
5.6 Conclusion and future works . . . 67

6 Statistical mentions from textual claims 69
6.1 Introduction . . . 69
6.2 Statistical claim extraction outline . . . 69
6.3 Entity, relation and value extraction . . . 72
6.3.1 Statistical entities . . . 72
6.3.2 Relevant verbs and measurement units . . . 73
6.3.3 Bootstrapping approach . . . 74
6.3.4 Extraction rules . . . 74
6.4 Evaluation . . . 76
6.4.1 Evaluation of the extraction rules . . . 76

6.4.2 Evaluation of the end-to-end system . . . 76
6.5 Implementation . . . 77
6.6 Related works . . . 77
6.7 Conclusion and future works . . . 78

7 Topics exploration and classification **81**
7.1 Corpus construction . . . 82
7.2 Topic extraction . . . 83
7.3 Topic classification . . . 84
7.3.1 Preliminaries . . . 84
7.3.2 Model training . . . 85
7.3.3 Evaluation . . . 86
7.4 Conclusion . . . 86

8 Conclusion **87**
8.1 Summary . . . 87
8.2 Perspectives . . . 88

Bibliography **91**

Chapter 1

Introduction

1.1 Motivation

The development of widespread popular information production and sharing infrastructure has lead to any individual having access to information quickly and easily through various means: social networks, forums, blogs, newspapers, etc. Anyone with access to the Internet is thus potentially a digital content producer. Although information is easy to access, it has become increasingly difficult for information consumers to assess the credibility of content found online. A fake news article with an eye-catching title or fake images could be shared instantly to thousands of people who may redistribute it, without verifying its misleading content. As a consequence, misinformation travels fast, and it may lead to dramatical consequences in real life[1].

Verifying the truthness of information is hard, even for professionals like journalists. This process could be considered as consisting of three tasks:

1. **Extracting *claims* ("a statement that something is true or is a fact"[2]) from the published information**: this is necessary, because not all the published content can be checked, even if infinite manpower was available. In particular, text stating sentiment, stance, emotion etc. is by definition not worth checking. In our work, we focus on *statistic claims* (Chapter 6), i.e., sentences containing relationship between a statistic entity, e.g., GDP (Gross Domestic Product), with a statistic value, e.g., 5%. We believe that this type of claim enable us to to tackle more complex claims in the future.

2. **Searching for relevant trustworthy data against which to check the claims**. We focus on the statistic spreadsheets and HTML tables since these data sources haven't been receiving much attention in the literature (see Section 3.2) but they contain highly relevant data for the fact-checking task. The difficulty of extracting meaningful information is the creativity of content providers, i.e, they can organize data as they see fit. Other type of data sources which have been well studied in the literature are search engine, knowledge base, Wikipedia, fact-checked claims (Section 3.2).

[1] As a high-profile example, the "Pizzagate" false rumour (https://en.wikipedia.org/wiki/Pizzagate_conspiracy_theory) has lead some outraged individual to actually open fire to revenge supposed victims of a (non-existent) child molester ring.

[2] https://dictionary.cambridge.org/dictionary/english/claim

3. **Assessing claim truthfulness by comparing the claim with the reference data.** There are many approaches to assess the truthfulness of claim in the literature (Section 3.4). We decided to let professional (e.g., journalist, fact-checker) perform this task with the relevant data that provide to them from the above task. From our point of view, it is really difficult to develop an automatic system because even human are not good enough at this task yet.

Sample reference data may come from statistic corpora established by trustworthy institutions, scientific publications, etc. The relevant data could be found in structured data in databases or unstructured data such as texts, images, etc.

We make the following observation here. Research efforts have been invested toward automatically computing a "reference database", such as a knowledge graph, by compiling a very large number of data sources and assigning them some credibility scores. In this thesis, guided by our discussions with fact-checking journalists from the Le Monde journal, we take a different approach. We make the assumption that journalists know what sources are to be trusted, based on their professional training for what constitutes a reliable source, as well as the professional experience they acquire. Thus, throughout this thesis, we consider that the reference data is known, and work in this "closed-world" model, focusing on the best exploitation possible of the available reference data.

Manually extracting claims, identifying relevant reference data sources and assessing claim truthfulness is very time consuming.

The starting point of the work performed in this thesis is the observation that content management techniques (database management, information retrieval, knowledge bases management, and natural language processing) have the potential to (at least partially) automate the process. This was the assumption underlying the ANR project ContentCheck[3], which provided the framework of the present thesis. Its goal has been to develop tools toward automatizing such fact-checking pipelines.

Specifically, focused on the following problems:

1. Improving the accessibility and usability of reference data sources.
2. Identifying facts from textual content, e.g., newspapers, that need to be checked.
3. Querying the relevant data from the database to verify a given fact in an efficient manner.

1.2 Contributions and outline

The manuscript is organized as follows.

Chapter 2 introduces a set of preliminary notions we build upon: the Resource Description Framework (RDF), and a set of notions pertinent to the area of Information Extraction.

Chapter 3 presents the state of the art in the fact-checking tasks which are: claim extraction, claim accuracy assessment, and reference source search. A set of related datasets, fact-checking challenges, and some end-to-end fact-checking systems are also presented.

[3] http://contentcheck.inria.fr/

In Chapter 4, we discuss in details our algorithm to extract RDF data from statistic spreadsheets. The task is challenging as spreadsheets do not have an homogeneous layout; further, the extraction needs to preserve the structural relations between cells in a table (e.g., a given cell corresponds to a given line header and a given column header). This is complicated by the frequent presence of merged header cells in statistic spreadsheets. Our contributions are:

- an algorithm to extract header and data cells from spreadsheets (Excel files), and
- an RDF corpus of the extracted data, which we release as Linked Open Data.

In Chapter 5, we develop a search algorithm to retrieve the relevant datasets with respect to a user query. The challenges are quantifying the relevance of datasets and identifying, within these datasets, the most relevant data snippets. Our contributions are:

- an efficient search algorithm;
- an evaluation against baselines confirming its interest, and
- a prototype available online for end users.

Chapter 6 presents our approach to identify statistical mentions from claims, and generate queries to be solved by the search algorithm presented in Chapter 5. We develop algorithms to identify whether a sentence contains a statistical claim and then extract the necessary information to formulate it. Our contributions are:

- an unsupervised approach for claim extraction, as well as
- the integration with the works described above, into an end-to-end system that identifies the reference information most relevant for a given claim found in an input text such as a media article or interview. For textual content from social networks (e.g., tweets), some more pre-processing steps should be performed to filter out the abbreviations, emoticons, etc.

In the above chapters, some related works specific to the techniques introduced in each chapter, but which did not pertain to the state of the art presented in Chapter 3, are included next to the most relevant material.

In a related project to fact-checking task, we developed a system which extracts topics from a text corpus (consisting of tweets and news articles), and classifies each text according to these emergent topics (Chapter 7). This system does not require labeled data for training; however, it obtains high performance in term of classification accuracy.

We conclude the thesis in Chapter 8. All the algorithms devised during this thesis are distributed as open source software.

Chapter 2

Preliminaries

In this chapter we present the notions of the Resource Description Framework (Section 2.1) and Information Extraction (Section 2.2). We conclude the chapter with the reason why we chose these techniques to solve our problems (Section 2.3).

2.1 Resource Description Framework

RDF (Resource Description Framework) is a framework for describing resources on the web. It is a W3C Recommendation from February 10th, 2004 [W3C, 2004].

The following concepts are used to define resources:

- **URI** (Uniform Resource Identifier) [Berners-Lee et al., 1998] is an unambiguous string that identifies a resource.
- **Literal** is used to identify values such as text or number.
- **Blank node** represents a resource for which an URI or literal is not given.

An **RDF triple** has three components: the *subject* which is an URI or a blank node; the *predicate* which is an URI that describes a binary relation between subject and object; the *object* which is an URI, a literal or a blank node. It is presented as `(subject, predicate, object)`. An **RDF graph** is a set of RDF triples.

A standard URI part of the RDF format allow to describe the **type** of RDF resources. For example, to specify that a `macbook` belongs to `Computer` class, we use the following statement

```
ex:macbook rdf:type ex:Computer .
```

RDFS (RDF Schema) provides a small set of standard URIs which can be used to state relationships between classes and/or properties. For instance, the RDFS `rdfs:subclassOf` URI can be used to specify specialization relationships, for instance:

```
ex:Macbook rdfs:subclassOf ex:Computer .
```

Similarly, RDFS provides the `rdfs:subpropertyOf` URI for stating that a property is a specialization (particular case of) another, for example `ex:isCapitalOf rdfs:subpropertyOf ex:isCityOf .` Finally, `rdfs:domain` allows stating that any resource having a certain

```
PREFIX ex: <http://example.com/exampleOntology#>
SELECT ?city
WHERE
  {
    ?x  ex:cityname       ?city      ;
        ex:isCityOf       ?y         .
    ?y  ex:countryname    ex:France  .
  }
```

Figure 2.1: Sample SPARQL query.

property is of a certain type (called a *domain* of that property), and similarly `rdfs:range` can be used to state that any value of a certain property belongs to a certain type (called *range* of the property). For example, `ex:City` is the domain of property `ex:isCapitalOf` and `ex:Country` is its range.

Open-World Assumption (OWA) It is important to understand that RDFS constraints are to be interpreted following an Open-World Assumption (they lead to implicit knowledge that was not explicitly present in the original RDF graph), not in a Closed-World Assumption (where all the data that holds is assumed to be part of the database). Under the Closed-World Assumption, a subtype constraint could be seen as "violated" if a resource of type c_1 was not also stated to be of type c_2, where c_2 is a supertype of c_1. In contrast, using RDFS constraints, there is never a "violation" or "inconsistency" (the constraint language is not expressive enough); RDFS rules only lead to more (entailed) triples.

SPARQL (SPARQL Protocol and RDF Query Language) [Prud'hommeaux and Seaborne, 2008] is an RDF query language. It is a declarative query language (like SQL) that could manipulate and retrieve RDF data. The sample SPARQL query in Figure 2.1 returns all cities (`?city` variable) of France. Using the concept of *basic graph pattern* (an RDF graph which has subject, predicate, or object replaced by a variable), SPARQL answer the `SELECT` query by returning a mapping of variables in the query with URIs and literals from the queried RDF graph.

2.2 Information extraction

2.2.1 Information extraction tasks

Web scraping or web data extraction is the process of fetching web pages and extracting from them certain kind of information. For instance, one can be interested in extracting from Web pages that show real estate ads, the kind of property being shown (a house, a flat etc.) and its other characteristics, such as surface, price etc. The extracted data is stored in a structured data format, e.g., JSON (Javascript Object Notation), CSV (comma-separated values), etc. or in a database for future use.

Given a query q (a string of characters) and a collection D of documents (where a document could be either a structured data collection, or non-structured data such as a text), **document ranking** is the task of finding the subset D_q documents of documents relevant to the query

q, and rank them in the descending order given by a *relevance score function*. For instance, given the query "age of Emmanuel Macron", and considering the set of Web pages indexed by a search engine as D, document ranking consists of finding the k (whose typical values are around 10 or so), which are most relevant for the given search terms. The principles, main data structures and algorithms for solving this task are described in textbooks such as [Frakes and Baeza-Yates, 1992].

Focused information retrieval is the task of locating relevant pieces of information in a document, in order to to answer a query. For instance, given the query "age of Emmanuel Macron", instead of a ranked list of Web pages, focused information retrieval would try to return a text snippet such as "Emmanuel Macron turned 40", found within a document.

Entity extraction (also known as named-entity recognition, entity identification, or entity chunking) is the process of identifying and classifying elements from text into pre-defined categories. For example, identifying all politician names from a collection of newspapers.

Text classification is the task of classifying a text unit (sentence, paragraph, article, etc.) into appropriate categories. An example of this task could be classifying newspapers' content into *economy*, *politics*, *technology*, etc. Fake news detection task is a text classification task which aims to classify whether a given text from newspaper is fake or not.

Stance detection is the task of detecting the semantic relationship (stance) between two pieces of text. E.g., the text "I could not focus well because of the notifications from Facebook" expresses a *support* stance toward the following text "The usage of social network reduces our concentration". Some works is covered in these sections 3.1, 3.4, 3.6.

2.2.2 Machine learning for information extraction

Recent years have seen an explosive adoption of *machine learning* techniques in the Information Extraction area. We distinguish the following family of techniques:

- **Supervised learning** designates a family of machine learning methods that map an input data point, i.e., a text representation (Section 2.2.5) of text unit (e.g., a phrase, a sentence, a paragraph or an article), to an output, after learning the input-output correspondence on a *labeled dataset*. A standard practice is dividing the labeled dataset into three parts: the *training set* of labeled examples is used to learn ("train") the statistical model, the *development set* (also called "dev set", in short) is used to tune the model's parameters, finally the *test set* is used to evaluate the model's performance. The advantage of supervised learning methods is that they are based on "gold standard" points (usually provided by humans); however, this is also their weakness, as human labor is generally expensive. Another weakness is the quality of gold standard because two humans could easily give two different labels to the same data point. Examples of supervised learning are
  - Linear Support Vector Machines (Linear SVM) [Cortes and Vapnik, 1995]
  - Gaussian Naïve Bayes [Friedman et al., 1997]
  - Random Forests [Breiman, 2001]
  - Logistic Regression [Cramer, 2002]
  - Gradient Boosted Decision Tree [Friedman, 2000]

- **Unsupervised learning** is a class of machine learning methods that do not require labeled data; this is an advantage given that they are no longer dependent on the availability of costly human-labeled examples. These methods try to learn from the data distribution in order to discover the interesting structures or semantic characteristics of the data. For example, given a dataset of news articles, a classifier following an unsupervised approach could group together similar articles thanks to the similarity of their semantic representations. Some examples of unsupervised learning are
  - k-Nearest Neighbour Classifiers [Cunningham and Delany, 2007]
  - Autoencoder [Rumelhart et al., 1986]
  - Local outlier factor [Breunig et al., 2000]
- **Semi-supervised learning** is a hybrid between supervised and unsupervised learning. Given a set of labeled data L and another (bigger) set of unlabeled data U, the semi-supervised learning methods try to combine the useful information from both sets in order to improve over the performance of supervised models trained on only L. Further, by relying both on L and on U, semi-supervised learning reduces the dependency on a large set of labeled examples. An example of this approach is Transductive SVM [Joachims, 1999].
- Another approach to overcome the difficulty of creating large labeled datasets is **distant supervision**, which leverages knowledge from a knowledge base to generate the training data. However, this method faces the challenge on handling potentially *noisy labeled (training) data* thus obtained. This learning paradigm was proposed in [Mintz et al., 2009].
- **Data programming** is an approach proposed by [Ratner et al., 2016] that could create labeled data from user's *labeling functions* using heuristics, patterns, etc. to express weak supervision signals. For example, on the task of extracting the relation *isSpouseOf* from text, a user could write the labeling function: "If two people who have the same child then they might be husband and wife" in order to return a label 1 (a positive training example) for the relation *(A. Nguyen, isSpouseOf, B. Tran)* from the sentence "C. Nguyen is the child of A. Nguyen and B. Tran". The authors implemented a system called Snorkel [Ratner et al., 2017] which learns from the agreements and disagreements of all the provided labeling functions to generate *probabilistic label* for every unlabeled data example. Their system analyzes the relationships of labeling functions to decide when to simply apply majority vote and when to infer the label from all the labeling functions. Finally, a supervised machine learning model is trained to minimize a loss function of the probabilistic labels. The authors develop connectors to popular machine learning libraries to let end-users have a wide range of platform choices.

  The predecessor of Snorkel is DeepDive [Zhang, 2015]. This system relies on heuristic rules and distant supervision to obtain the training data. Multiple labels for the same data example are resolved by majority vote.

2.2.3 Deep learning for information extraction

We recall here basic terminology and most frequently recurring deep learning models to which this manuscript will refer in the sequel.

A **neuron** is the basic unit (a *node*) of a neural network. A neuron receives its inputs consisting of a vector of numerical values $\{x_1, x_2, \ldots, x_n\}$, a vector of weights $\{w_1, w_2, \ldots, w_n\}$ that reflects the importance of each x_i, and a bias vector b. The output of neuron Y is computed by a non-linear activation function f:

$$Y = f(\sum_{1 \leq i \leq n} w_i \times x_i + b)$$

The purpose of f is learning the *non linear* representations of the input data.

The most simple neural network is a **feed-forward neural network** that consists of multiple **layers**. Each layer is a collection of neurons. There are **edges** connecting neurons from two adjacent layers. Each edge contains weight that has been discussed previously. Layers could be classified into three types: *input layer* that represents the input data, *hidden layer* that transforms the data from the input layer to the output layer, *output layer* that represents the expected output data. A feed-forward neural network with more than one hidden layers is called **multi layer perceptron**.

There are more sophisticated neural network architectures. **Convolutional neural networks (CNNs)** [LeCun and Bengio, 1998] are designed to classify images. A CNN uses **convolutional layers** to learn the representations of local regions (e.g., a pixel and its eight surrounding pixels) from the input image. Each convolutional layer learns a specific visual feature. All these extracted features are sent to the output layer for the classification task. To handle text data, a local region is considered as n contiguous words. Specific linguistic feature is learnt in each convolutional layer. CNNs have been applied in text classification [Kim, 2014], relation extraction [Zeng et al., 2014], etc.

Recurrent neural networks (RNNs) are designed to process **sequence data**, e.g., text as a sequence of words. In the standard feed-forward neural network, neurons in the hidden layer are only connected to neurons from the previous layers. RNN allows the connection between two adjacent neurons in the hidden layer. This modification makes the network learn from the *history* of the sequence encoded in the hidden layer. The effectiveness of RNNs has been shown in many tasks such as text classification, time series prediction, etc. RNN variants such as Long Short Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997], Bidirectional LSTM (Bi-LSTM) [Schuster and Paliwal, 1997] are more popular than the vanilla RNN in practice.

2.2.4 Metrics for evaluating information extraction quality

It is often the case that we need to evaluate the *quality* of an information extraction process, in order to get a quantitative grasp of the trust which can be put in its output.

To evaluate the quality, a *gold standard* of answers considered correct (typically provided by humans) is usually assumed available. The gold standard is in some cases a *set*, e.g., objects which are sure to belong to a certain class (in a classification problem), or the set of all mentions of humans which an information extraction algorithm must identify in a text. In other cases, the gold standard is a *list*, for instance, when the problem is to return a ranked list of answers from the most relevant to the least relevant, and a ranked list of relevant answers is specified by a human. Based on such a gold standard, for a given information extraction method which returns a certain set of answers to a given task, a set of popular metrics are:

- *Precision*, denoted p, is the fraction of the returned results that are part of the gold standard. Precision can be seen as reflecting the usefulness of a method, i.e., how many of the correct results are returned.

- *Recall*, denoted r, is the fraction of the gold standard that is part of the returned results. Recall can be seen as reflecting the completeness of the method. Precision indicates the usefulness while recall indicates the completeness.

- There is a natural tension between precision and recall; returning more results cannot decrease precision, but it can decrease recall, and vice versa. Thus, a single metric including both is the *F1-score*, defined as the harmonic mean of the two previous metrics: $F1 = 2 \times \frac{pr}{p+r}$.

- The above discussion is based on a "binary" setting where a result can be either be part of the gold standard (e.g., be "relevant") or not. In a more general setting, e.g., classification with more than two classes, two variants of the F1-score can be defined, respectively, are *macro-average* and *micro-average*. The macro-averaged F1-score is the average of the F1-score of all classes. The micro-averaged F1-score is the weighted average of the classes' F1-scores, which takes into account the contribution of all classes.

- *ROC curve* (receiver operating characteristic curve) is a plot of the two metrics of a classification model: recall on the vertical axis, and *FPR* (False Positive Rate) in the horizontal axis. FPR is calculated as *FPR = False Positive / (False Positive + True Negative)* where False Positive refers to the situation when a binary classifier incorrectly predicts the positive class and True Negative refers to the case when a binary classifier correctly predicts the negative class. The curve is constructed from all pairs $(recall, FPR)$ corresponding to all classification thresholds, i.e., a float value to compare with the binary model's probability in order to classify an example as positive or negative class.
  Area Under the ROC Curve (AUC) is the area below the ROC curve which indicates the probability that a random positive example could be ranked higher than a random negative example.

The above metrics apply to a set (or unordered list) of results. For the context where the order of the retrieved results is important, for example web search results, we have to use the following metrics:

- *Precision at k (P@k)* is the precision computed based on the top-k retrieved results.

- *Average precision at k* is defined as as $\frac{1}{min(m,k)} \sum_{i=1}^{k} P@i \times rel(i)$ where m is the number of actual relevant results, $rel(i)$ indicates if the ith result is relevant ($rel(i) = 1$) or not ($rel(i) = 0$).

- *Mean average precision at k (MAP@k)* is the average of n average precision at k. For example we execute $n = 100$ different queries against a search engine, compute the average precision at $k = 10$ for each query, and then compute $MAP@k$ to evaluate the performance of this system.

2.2.5 Text representation

In order to apply machine learning techniques (Section 2.2.2) on plain text, the latter first needs to be represented as numerical vectors.

A **language model** computes the probability of words that could appear in a fixed length sequence. An **n-gram**, is a contiguous sequence of n words. For example the sentence "Machine Learning for Information Extraction" is represented by the following sequence of 2-grams: "Machine Learning", "Learning for", "for Information", "Information Extraction". A unigram and a bigram are n-grams where n equals to 1 and 2 respectively. An **n-gram** language model uses the sequence of n previous words in order to predict the next word.

In recent years, Natural Language Processing in general and Information Extraction in particular widely take advantage of **word embeddings** which is a family of techniques capable of computing, starting from words/phrases, vectorial (multidimensional) representations thereof, in a numerical space. These techniques are based on the distributional hypothesis "a word is characterized by the company it keeps" (popularized by [Firth, 1957]), and they produce *dense* vectors (many non-zero components), whereas BoW produces sparse vectors (also called "one-hot", with one or few non-zero components). A method representative of the word embeddings class is word2vec [Mikolov et al., 2013]. The interest of word embeddings is that once a text can be reduced to such a numerical representation, proximity (or similarity) between two texts can be computed easily, e.g., by means of a scalar product of the two. Word embeddings are typically learned from large, unlabeled text corpora. There are two variants of word2vec: Common Bag Of Words (CBOW) and skip-gram. CBOW predicts the target word w in a given sequence, e.g., "Have a w evening!", using the *context* of w ("Have a ... evening!") of w as input. On the contrary, skip-gram predicts the context using a given word as input. Both of these variants are based on training a neural network on text corpora. According to [Mikolov et al., 2013], skip-gram works well with small amounts of the training data, and represents well words or phrases which are rare, while CBOW is several times faster to train than the skip-gram, and has slightly better accuracy for the frequent words.

The success of word2vec inspired NLP (Natural Language Processing) researchers to come up with more powerful models such as

- Glove [Pennington et al., 2014] computes the embeddings from the statistics of word co-occurrences in a given corpus.
- fastText [Joulin et al., 2016] learns the embeddings from n-gram of characters, e.g., character 3-grams of the word "fast" are "fas" and "ast". This approach helps to deal with out-of-vocabulary words better.
- ELMo [Peters et al., 2018] introduces *contextualized embeddings*, i.e., the same word would have different vectors in different contexts.
- ULMFiT [Howard and Ruder, 2018] introduces a fine-tuning technique to obtain the meaningful text representation with each specific task such as text classification.
- BERT [Devlin et al., 2018] introduces the "masked language modeling" technique to take into account both left and right contexts simultaneously.

They also proposed similar techniques for representing phrases and sentences in a multidimensional space, including for instance Doc2Vec [Le and Mikolov, 2014].

In our work, we rely on word2vec in order to obtain a list of words that are semantically similar to a given keyword. This list helps our search engine (see Chapter 5) to obtain more informative matches instead of exact matches.

2.3 Conclusion

One problem that we solve in this thesis is retrieving relevant statistic data with respect to a factual claim from text. The data comes from spread files which contains relationship between numerical data cell and its corresponding pair of header cells. There is also the parent-child relationship between a pair of header cell. The RDF framework allows us to easily model these relationships.

We implemented Information Extraction techniques in order to retrieve relevant data from text. Inspired by the success of Deep Learning in these recent years, we experimented with it as described in Section 8.1. The empirical results were not good as we expected and we decided to adopt "white-box" approaches, i.e., computing relevant score function of a dataset (Chapter 5) and defining extraction rules from a given sentence (Chapter 6).

Chapter 3

State of the art of computational fact checking

Journalists have been cooperating with professionals from other fields such as computer science and statistics to process numerical data, produce and distribute information. **Data journalism** [Gray et al., 2012] is the term to refers to that cooperation. With more and more digital contents produced, people encounter the rise of **fake news** which is the false, made-up, or misleading information with the aim of spreading to large (online) audiences. These news make use of the appealing imagery to attract more readers while presenting untrustworthy and unreliable sources [Machado et al., 2019].

The combat against fake news also attracts the attention of many researchers on the task of ***posteriori* fact checking** which is defined by [Cazalens et al., 2018] as the process of:

1. extracting claims from some discourse,
2. searching for the facts the claims are based on,
3. assessing the accuracy of the claim with regards to those backing facts, and
4. providing perspective to claims for which there is no straightforward settlement.

This process is illustrated in Figure 3.1. In another study, [Graves, 2018] defined the automated fact checking process as a loop of three sub-tasks: identification, verification, and correction (Figure 3.2). Fact checking is normally performed by trained **fact checkers** by evaluating previous speeches, debates, legislation and published figures or known facts [Thorne and Vlachos, 2018]. [Rivas, 2019] propose that fact checking could be done using less resources with crowdsourcing: the crowd present different arguments and evidence to verify a claim and a group of moderators will aggregate all the provided information to provide the final fact-check. However the time consuming steps such as argument classification and stance detection should be automated. Finally, fact-checks should be published in order to correct the spread of fake news and untrustworthy claims [Graves, 2018].

In this chapter, we will firstly consider different methods for extracting claims (section 3.1). In Sections 3.2, 3.3, we will explore which data sources could be used for performing the fact-checking task. The different techniques to verify the truthfulness of a given claim are presented in Section 3.4. Some fact checking related challenges are mentioned in Section 3.5. Finally, we will mention some representatives of automated end-to-end fact checking systems

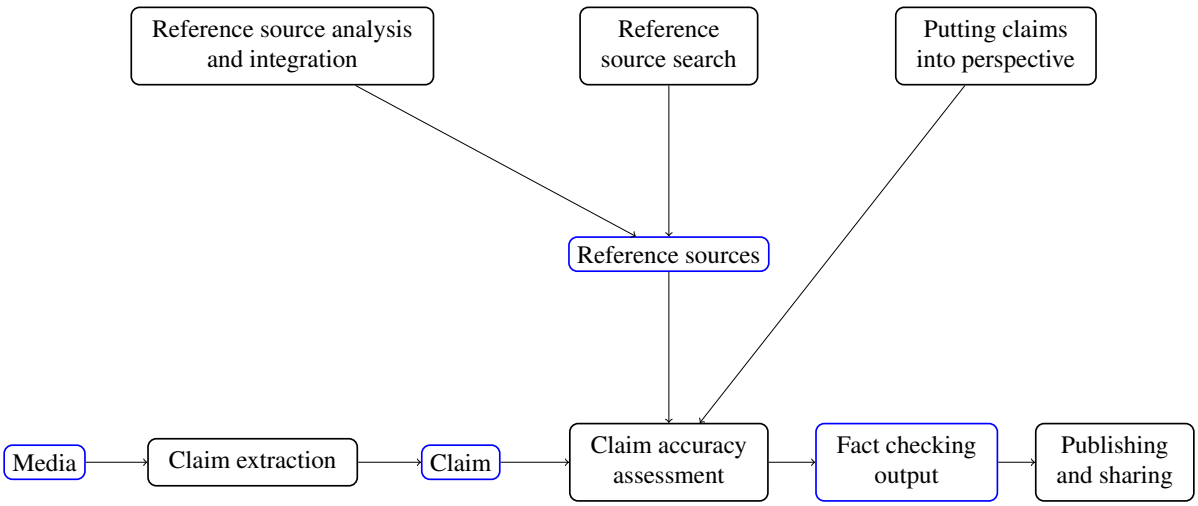

Figure 3.1: Fact checking tasks [Cazalens et al., 2018]

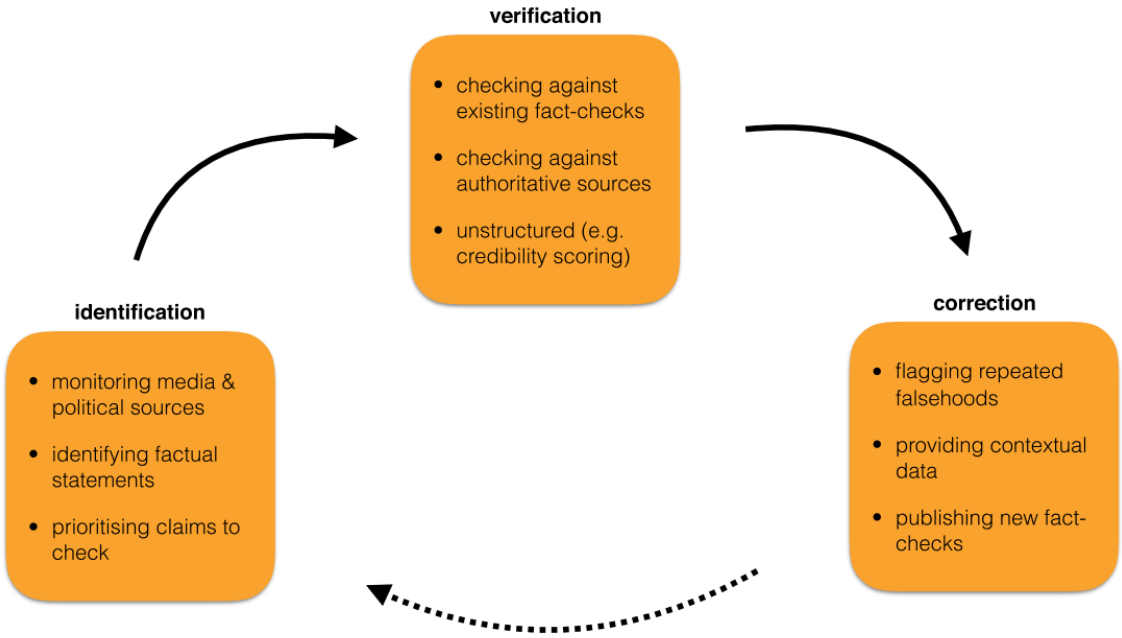

Figure 3.2: Core elements of automated fact-checking [Graves, 2018]

in Section 3.6 and end this chapter with a conclusion in Section 3.7.

3.1 Claim extraction

Two broad classes of methods can be used for this problem. Unsupervised methods are based on topic modeling algorithms, or sentence ranking; we discuss such methods in Section 3.1.1.

In a supervised setting, claim extraction is framed as text classification task, for example classifying the given text (a sentence, an article) as "claim" (e.g., "The president claimed that the unemployment rate of France has been going down in these three years.") or "not claim" (e.g., "The weather is nice today."). It relies on labeled data, usually obtained through crowdsourcing. A supervised machine learning model is then trained using a set of appropriate features or input representation. We cover these techniques in Section 3.1.2).

3.1.1 Unsupervised approaches

Pattern matching In [Levy et al., 2017], claims are extracted from Wikipedia using pattern matching with an automatically generated lexicon, and developing a *ranking score* on the matched sentences. We detail this below.

Firstly, the authors manually identify a list of debate topics (main concepts), denoted MC. Each of these topics is a concept, e.g., "doping in sport", "boxing" etc. From Wikipedia, the authors extract a corpus of 1.86 millions sentences that contain these topics.

A sentence which has the structure "... *that* ... **MC** ..." is a candidate in which they search for a claim, e.g., "The president claimed *that* his government has reduced the unemployment rate to 5%." The corpus is divided into two sets: c_1 which contains the above structure and c_2 which does not. They define a **sentence suffix** as the part of the sentence that follows a topic from MC. Then, for each word w in the vocabulary obtained from all sentences, they compute $P_{suff}(c_1|w) = n_1/(n_1 + n_2)$, where n_1 is the number of sentences in c_1 that contain w in the sentence suffix, and n_2 is the number of sentences in c_2 that contain w. If $P_{suff}(c_1|w) > \frac{|c_1|}{|c_1| + |c_2|}$, then w belongs to the set of words called *Claim Lexicon (LC)*.

To rank a sentence s that matches the pattern "... that ... MC ... cl ...", where cl is a word in CL (if there are more than one word from CL that appears in s after MC, the first one is selected), they take the average of two following scores:

- *w2v*: for each word w in s following the first 'that', they find the best cosine similarity between the word2vec representation of w and that of each word in MC, and then average the obtained scores;
- *slop*: this is number of tokens between the word 'that' and cl.

In the above, $w2v$ is an average of two cosine similarities (thus, it is between 0 and 1) while $slop$ is a natural number, thus it can go above 1. Averaging them amounts to (strongly) penalizing high $slop$ values.

From a highly ranked sentence, a claim is extracted as being the words appearing after the word 'that', and until the end of sentence. In the sample sentence: "The president claimed *that*

his government has reduced the unemployment rate to 5%.", the extracted claim would be "his government has reduced the unemployment rate to 5%".

Their system reports a P@5 (precision at 5, recall Section 2.2.4) score of 0.31 on the development set, and 0.32 on the test set. While these figures are quite low, the authors show that they improve over those of prior supervised extraction approach [Levy et al., 2014].

Topic modeling and topic extraction Another unsupervised approach to extract claim uses topic modeling algorithms to characterize the text that potentially contains a claim. [Ferrara et al., 2017] relies on Hierarchical Dirichlet Processes [Teh et al., 2004] in order to find the optimal number of topics from a text corpus. From the generated topics distribution, they compute a score called *attraction score* and then use it to classify sentences into four categories: *claim* which states that something is the case without any further proof, *premise* which is the proof to support or attack a claim, *major claim* which is the claim present in the beginning (introduction) of a document in order to express the main semantics of the whole document, and *non-argumentative* which is text that does not belong to the previous three categories. They report an accuracy of 0.3, which outperforms the accuracy of 0.1 of a random baseline. This choice of the baseline can be questioned; the reported accuracy of this method is quite low. In the error analysis, the authors gave an example of non-argumentative sentence "Why do some parents not think their kids can attain?" which has been labeled as premise by the model. They attempt to solve this problem in future work by studying the dependency relations among sentences in text.

Similarly, [Sobhani et al., 2015] use Non-Negative Matrix Factorization (NMF) [Lee and Seung, 2000] to extract topics from a text corpus which is a collection of 1,063 comments on 8 articles discussing breast cancer screening. Each topic is represented by a list of keywords, e.g., the topic "Mammo can cause cancer by its radiation" is represented by "radiation, lumpectomy, expose, need, colonoscopy, surgery, chemo, cause, radiologist, machine, treatment, exposure, safe, thermography" [Sobhani et al., 2015]. Annotators attach to each of these topics some arguments from a predefined list, such as "Mammo may cause cancer", "Financial benefit of mammo for medical industry", etc.

On a corpus of 781 comments on news articles, they report an F1-score of 0.49 for the topic classification task (classifying the arguments to which a comment belong). Finally, on the stance classification task (classifying each comment as *for* or *against* the comment's arguments), they train an SVM model using:

- TF-IDF [Jones, 1972] features: each comment is transformed into a TF-IDF word count vector;
- the predicted topic topics from NMF

which lead them to an F1-score of 0.77.

3.1.2 Supervised methods

SVM classification The authors of [Liebeck et al., 2016] seek to annotate a corpus with three categories:

1. *major position*, which is a citizen suggestion (e.g., "We should build a playground with a sandbox."),
2. *claim*, which expresses a "for" or "against" stance toward a major position (e.g., "I dislike your suggestion."), and
3. *premise*, which is a reason to attack or support a major position (e.g., "This would allow us to save money.").

Each sentence is also annotated as *argumentative* (containing argument components) or *non-argumentative*. Finally, they obtain a dataset that consists of 548 major positions, 378 claims, and 1,244 premises. The authors observe that comment writers use *different tenses and sentence structures* for each category, e.g., claims are often stated with "I agree!", as in: for "I agree! Building the airport would create more jobs". Based on this observation, they use the frequency of POS[1] tags and dependencies in the TIGER annotation scheme [Brants et al., 2004] as features of the SVM classifier. They also use the unigrams and bigrams of the sentences as binary features (values of 1 or 0 if the unigram/bigram appears or does not appear respectively in the given sentence).

On the subtask of classifying sentences as argumentative or non-argumentative, they obtained a macro-averaged F1-score of 0.69. On the subtask of classifying sentences into the three categories (major position, claim, premise), they got a macro-averaged F1-score of 0.68.

SVM is also used by [Lippi and Torroni, 2016] to detect claims from political speeches. They extract the audio features using the RastaMat library[2] and combine them with standard text features: unigrams, bigrams, part-of-speech tags[3], and lemmas[4] sentences.

On three datasets of 122, 104, and 160 audio samples, respectively, they report the 10-fold cross validation macro-averaged F1-score of 0.52, 0.52, and 0.31 respectively.

Logistic regression [Konstantinovskiy et al., 2018] rely on InferSent [Conneau et al., 2017] which takes word order into account to represent the input sentence. In this collaborative work with Full Fact[5], they define seven categories to annotate input sentences:

- personal experience (e.g., "I can't save for a deposit."),
- quantity in the past or present (e.g., "The Coalition Government has created 1,000 jobs for every day it's been in offic",
- correlation or causation (e.g., "Tetanus vaccine causes infertility"),
- current laws or rules of operation (e.g., "The UK allows a single adult to care for fewer children than other European countries."),
- prediction (e.g., "Indeed, the IFS says that school funding will have fallen by 5% in real terms by 2019 as a result of government policies."),
- other type of claim (e.g., "The party promised free childcare"),
- not a claim (e.g., "Questions to the Prime Minister!").

[1] the category of words such as noun, adjective, verb, etc.
[2] http://labrosa.ee.columbia.edu/matlab/rastamat
[3] the category of words such as noun, adjective, verb, etc.
[4] the canonical form of a set of words, e.g. *work* is the lemma of *work, works, worked, working*.
[5] https://fullfact.org

According to Full Fact's experience, these categories are sufficient to cover texts from political TV shows. Then 80 volunteers were recruited from Full Fact's newsletter to annotate a dataset consisting of 6,304 sentences using Prodigy[6]. They consolidate these through a majority vote, to obtain a set of 4,777 annotated sentences where at least three out of five annotators agree on a sentence label. They expand the dataset by adding a set of annotated claims from Full Fact's database. The final dataset has 5,571 sentences.

After obtaining the sentence embeddings from InferSent, they concatenate it with the count of parts of speech and named entities found in each sentence to create the feature vector. They feed these vectors into standard supervised classifiers including Logistic Regression, Linear SVM, Gaussian Naïve Bayes, and Random Forests. Through their experiments of classifying a given text into "claim" and "not claim" categories, Logistic Regression has the highest performance in term of F1-score (0.83). This score is higher than those of ClaimBuster [Hassan et al., 2017] (0.79) and ClaimRank (0.77) [Gencheva et al., 2017]. The authors believe that ClaimBuster has the caveat of having to select a cut-off score to determine whether a given text is claim or not. The text representation in ClaimBuster is TF-IDF which is less informative in compared with the sentence embeddings representation in this work. They also discover that the POS/NER features do not contribute to their model's performance.

On the task of classifying claims into the seven predefined categories, their model achieves a micro-averaged F1-score of 0.7 and a macro-averaged F1-score of 0.48. The low macro-averaged score is explained by small number of examples of classes like "Current laws" and "Correlation or causation".

Recurrent neural network The approach proposed in [Hansen et al., 2019] is to develop a neural network for the task of finding check-worthy sentences, which takes advantage of large amounts of unlabeled data. Each word is represented by word embeddings to capture its semantics and by syntactic dependency tag[7] to capture its role, for example being the subject of its sentence. By using the syntactic dependency tags, they aim to discover the discriminative sentence structures for the check-worthy sentences. Given a pair of sentences s_1 and s_2, let

- W be the set of words that appear in both s_1 and s_2
- $t(w, s)$ be the syntactic dependency tag of the word w in the sentence s
- $overlap(w, s_1, s_2) = 1$ if $t(w, s_1)$ equals to $t(w, s_2)$ otherwise $overlap(w, s_1, s_2) = 0$

They define the overlap of the syntactic dependency tags of s_1 and s_2 as:

$$O(s_1, s_2) = \sum_{w \in W} overlap(w, s_1, s_2)$$

From three types of sentences: check-worthy sentences, non check-worthy sentences, and mixed of both, they sample randomly 10 pairs from each type and then compute the average $O(s_1, s_2)$. After repeating this experiment 1,000 times, they obtain the highest average overlap from the pairs of check-worthy sentences and conclude that syntactic dependencies could help to distinguish between check-worthy sentences and non check-worthy ones. The word representation consisting of word embeddings and syntactic dependencies is sent to a recurrent neural network. They use the attention mechanism [Bahdanau et al., 2015] in order to

[6] https://prodi.gy/
[7] https://spacy.io

Y	$\tilde{Y}$	Sentence
1	0.96	america has spent approximately six trillion dollars in the middle east , all this while our infrastructure at home is crumbling .
1	0.95	today , our total business tax rate is 60 percent higher than our average foreign competitor in the developed world .
1	0.26	its the same reason why she wo nt take responsibility for her central role in unleashing isis all over the world .
1	0.22	we will follow two simple rules ; buy american and hire american .
0	0.04	millions of democrats will join our movement , because we are going to fix the system so it works fairly and justly for all americans .
0	0.05	i have no patience for injustice .
0	0.94	in the last eight years , the past administration has put on more new debt than nearly all of the other presidents combined .
0	0.95	our trade deficit in goods with the world last year was nearly $ 800 billion dollars .

Figure 3.3: Model's prediction ($\tilde{Y}$) vs. ground truth labels (Y) [Hansen et al., 2019]. The intensity of the color increases with the relevance of the highlighted text.

identify the relevance of words/phrases with respect to model's prediction result. They propose a visualization of the model's prediction result, as illustrated in Figure 3.3.

The evaluation dataset E of [Hansen et al., 2019] consists of 2,602 sentences with check-worthiness annotations at sentence level. They further divide E into a training set $Train_E$ and a test set $Test_E$. The *weakly labeled dataset* W consists of 37,732 sentences from the public speeches of Hillary Clinton and Donald Trump during the 2016 U.S. election. They use the ClaimBuster [Hassan et al., 2017] API[8] to obtain a check-worthiness score s (a value between 0 and 1) for each sentence. Let f be the percentage of check-worthy sentences from $Train_E$, they find a threshold τ such that the percentage of sentences from W that have score $s \geq \tau$ is f. Sentences from W whose score $s \geq \tau$ are labeled 1 (check-worthy); the others are labeled 0 (non check-worthy). Using the labels thus obtained, they can train their neural network with a large amounts of data without labelling them manually.

15,059 documents[9] related to all U.S. elections (such as speeches, press releases, etc.) are used to train *domain specific word embeddings*. Using MAP, P@5, P@10, P@20 (see Section 2.2.4), and P@R (where R is the number of check-worthy sentences in the evaluation dataset) as evaluation metrics, the authors report significant higher scores (from 9 to 28%) compared to the following systems: ClaimBuster [Hassan et al., 2017], TATHYA [Patwari et al., 2017], ClaimRank [Gencheva et al., 2017] and the systems of [Zuo et al., 2018] (see Section 3.5), [Konstantinovskiy et al., 2018]. Incorporating the weakly labeled dataset W as an additional

[8] https://idir-server2.uta.edu/claimbuster/

[9] https://web.archive.org/web/20170606011755 and /http://www.presidency.ucsb.edu/

source of training data further raises all the scores.

3.2 Reference source search

For a given claim, a fact checking system searches for relevant reference sources from different sources. Below, we list the main categories of reference sources, and outline how the most relevant data for a given claim is found.

- **Search engine** such as Google, Bing. The claim could be issued directly against the search engine [Zhi et al., 2017]. Or it could be converted into search query by retaining only its verbs, nouns and adjectives [Nadeem et al., 2019, Karadzhov et al., 2017]. Named entities, e.g., location, person's names, etc. could also be added to the query issued to the search engine [Karadzhov et al., 2017, Wang et al., 2018].
- **Knowledge bases** such as DBpedia [Lehmann et al., 2015] and SemMedDB [Kilicoglu et al., 2012] can be leveraged to find the most probable paths in the knowledge base that connect the subject and object of a claim given in a triple format *subject, predicate, object* [Shi and Weninger, 2016]. The evidence facts related to a given claim could also be extracted from knowledge bases [Ahmadi et al., 2019].
- **Wikipedia pages** could be used to support or refute a given claim [Thorne et al., 2018]. A subset of sentences from these pages could also be retrieved to give specific evidence to explain the systems' decision.
- Previously **fact checked claims** could be compared with the given claim to find out whether a fact check for this claim already exists [Hassan et al., 2017, Lotan et al., 2013]. Such a comparison can be made based on a text similarity measure between the claim and the previously fact-checked claims.
- **Social media content** has been used as background (reference) information in [Goasdoué et al., 2013]: social media content is archived, then person names are used as search terms in order to identify the posts from a given actor.
- **Table cells** could be aligned with textual mentions of quantities in [Ibrahim et al., 2019].

3.3 Related datasets

We list below a set of datasets released by different organizations and/or research teams having worked in the above-mentioned areas.

- Wang [Wang, 2017] released a dataset[10] consisting of 12,836 human labeled short statements from the API of politifact.com[11]. Each data point includes a statement in plain text, a label which expresses the truthfulness of statement ("pants on fire", "false", "barely true", "half-true", "mostly-true", and "true"), the speaker (a politician) name and some

[10] https://www.cs.ucsb.edu/~william/data/liar_dataset.zip
[11] http://static.politifact.com/api/v2apidoc.html

meta-data (job title, party affiliation, state, and "credit history", which counts the different labels assigned to past statements of this speaker), and a context (venue/location of the statement, e.g. TV debate, social network, etc).

- The Fake News Challenge [Pomerleau and Rao, 2017] provided a dataset of 49,972 triples[12] of the form: news article's headline, news article's body text, and stance of the body relative to headline.
- The FEVER challenge [Thorne et al., 2018] introduced a dataset of 185,445 claims carrying a label among *Supported*, *Refuted* or *NotEnoughInfo*. For the first two labels, they also provided the combination of sentences as evidence to support or refute the claim.
- Datacommons.org released a dataset[13] which follow the ClaimReview[14] standard. It contains 12,545 fact checking articles, authored by human writers, as well as a set of extra information for each of them: details about the claim, the URL of fact checking article, the fact checker, the truthfulness rating, and the published date.
- [Popat et al., 2017] provided a dataset[15] of 4,856 claims from the Snopes website[16] together with their labels (true or false). They also provided the set of relevant articles collected from Google's search results for the given claims.
- [Mukherjee and Weikum, 2015] introduced the NewsTrust[17] dataset[18], consisting of 47,565 news articles from sources like New York Times, TruthDig, etc. Overall, the articles come from 5.6K distinct sources. Each article contains at least one review and rating from community members who are professional journalists and content experts. Community member profiles containing their occupation, expertise, demographics, reviews, and ratings are also included in this dataset.

3.4 Claim accuracy assessment

In order to verify a given claim, external sources could be retrieved to obtain more relevant data (Section 3.4.1). When the data is structured as a knowledge graph, claim accuracy assessment is related to the problem of *link prediction* or reasoning on logic rules (Section 3.4.2). Some other researchers rely on linguistic features of text to assess claim truthfulness (Section 3.4.3) or user inputs to verify a claim's credibility (Section 3.4.4).

3.4.1 Using external sources

[Karadzhov et al., 2017] use search engine results in order to verify the truthfulness of a claim expressed in natural language.

[12] https://github.com/FakeNewsChallenge/fnc-1
[13] https://www.datacommons.org/static/fact_checks_20190315.txt.gz
[14] http://schema.org/ClaimReview
[15] http://resources.mpi-inf.mpg.de/impact/web_credibility_analysis/Snopes.tar.gz
[16] snopes.com
[17] newstrust.net
[18] http://resources.mpi-inf.mpg.de/impact/credibilityanalysis/data.tar.gz

Given a claim in plain text, they rank its words by TF-IDF [Jones, 1972]. Only verbs, nouns and adjectives are kept. They also detect named entities, e.g., location, person names etc. using IBM's Alchemy API[19]. For example, from the original claim *"Texas, teenager Ahmed Mohamed was arrested and accused of creating a hoax bomb after bringing a home-assembled clock to school"*, they generate a query *"Texas, Ahmed Mohamed, hoax, bomb, clock, arrested, accused"*. The aforementioned approach to generate queries from claims is also adopted by [Nadeem et al., 2019]. But apart from the relevant documents identified through a search engine call, they also collect those from Apache Lucene[20] on the 2017 Wikipedia dump. Finally, a re-ranking model [Lee et al., 2018] is applied to obtain the top-k relevant documents.

The search query is executed against search engines (Google and Bing) and the ten first web pages returned by each of these are kept as *supporting documents*. From each such document, each sequence of three consecutive sentences is compared with the claim through a similarity function. The sequence most similar to the claim is called the *best-matching snippet*.

To generate a vector representation of a text, they transform the text into the average of the Glove embeddings (Section 2.2.5) of its words, which they subsequently pass to Bi-LSTM (Section 2.2.3). Five different texts are used to generate the vector representation of a given claim: the claim itself, Google supporting documents, Bing supporting documents, Google best-matching snippets, and Bing best-matching snippets. The similarity scores (cosine similarity of the embeddings vectors) between the claim and a snippet, or the claim and a web page are added to the concatenation of these vectors to form the final text representation. They pass this representation to a binary classifier to classify the claim as true or false.

On a dataset consisting of 761 claims from snopes.com, the above mentioned model achieves an F1-score of 0.772 on the test set.

Similarly, given a fact-checking article, [Wang et al., 2018] retrieves related documents using a search engine. Then a binary classifier is trained to predict the relevance of a document with respect with the given claim in the article. They train a model to classify the relevant document's stance (contradicting or supporting) with respect to the claim. To obtain the training data, they build a corpus of fact-checking articles containing exactly one ClaimReview[21] markup. They retain only articles from the highly reputed fact checking communities; this leads to a corpus of 14,731 articles.

To generate the relevant documents, they issue search queries to Google and collect the top-100 results. Queries of a claim are generated from:

1. The title of the fact-checking article and the claim of the ClaimReview markup.
2. The entities extracted from title and claim text using entity resolution [Mendes et al., 2011]. For example, given the claim "A video documents that the shootings at Sandy Hook Elementary School were a staged hoax", the extracted entities are *video*, *documents*, *shootings*, *Sandy Hook Elementary school*, and *hoax*. A confidence score is also given for each entity. This step is performed to retain only the important words in the query.
3. The 50 most popular search queries that led to clicks on the fact-checking article.

[19] www.ibm.com/watson/alchemy-api.html
[20] https://lucene.apache.org
[21] https://schema.org/ClaimReview

This approach has a recall of 80%.

Through crowdsourcing, they obtain a labeled corpus of 8,000 (fact-checking article, related document) pairs. The features of this model consisting of:

1. **Text similarity:** Each piece of text is represented by the weighted sum of word2vec embeddings over its words and phrases. They denote $sim(t_1, t_2)$ the cosine similarity of the text representations of two texts t_1 and t_2. A set of similarity scores are computed as following:
   - *sim(claim, document title)*
   - *sim(claim, document headline)*
   - *max sim(claim, each sentence from the document)*
   - *max sim(claim, each paragraph from the document)*
   - *max sim(s_a, s_d)* where s_a is a sentence from the article title, article headline, or from the article's sentences such that *sim(claim, s_a)* is above a pre-defined threshold θ and s_d is a sentence from the document
   - *sim(claim, document)*
   - *sim(claim, article)*
2. **Entity similarity:** From the list of the extracted entities from the article and the document, they build an *entity confidence score vector* $\{c_1, c2, \ldots, c_k\}$ where c_i is the confidence score of the extracted entity e_i if e_i appears in both sides, and c_i is 0 if e_i is found in only one side.
3. **Publication order:** The publication dates of the article and the document are collected. This information is collected since it is assumed that fact-checking articles are published around the same publication time of the claim.

Based on these features, they build a gradient boosted decision tree model to determine the relevance of a related document toward the fact-checking article. Their system got an accuracy of 81.7%.

Their idea for building the stance classifier is to identify within a documents, the elements that are contradictory to the claim. A contradiction vocabulary is collected for this purpose. Then they collect *key texts* including title, headline and important sentences whose similarity score with the claim is above a certain threshold. They call *key component* the concatenation of a key text, the sentence preceding it and the sentence after it. Finally, they extract unigrams and bigrams from key components and output a weighted n-grams vector over the contradiction vocabulary as feature of the gradient boosted decision tree model. On a labeled dataset of 8,422 examples, they obtained the accuracy of 91.6% on test set.

[Zhi et al., 2017] also rely on relevant news articles from Google search to verify a given claim from *snopes.com*. The sources credibility of these articles is taken into account. They implemented a system consisting of four modules: stance extraction, stance classification, article classification, and source credibility assessment, as follows:

- The *stance extraction* module retrieves articles relevant for the given claim through Google search (they retrieve the top 30 web pages). They generate snippet candidates by

removing sentences of less than K words, and combining each L consecutive sentences into a snippet. Through empirical studies, they find the optimal values $K = L = 3$. Then they compute the Doc2vec cosine similarity between the claim and each snippet, to retain the candidate snippets having a similarity higher than 0.55.

- The *stance classification* module trains a Random Forests classifier on the embeddings of candidate snippets and the labels from the fact database. This module outputs two stance scores, as the probability of having a snippet support or refute a claim.
- The *article classification* module trains a classifier (also using Random Forests) to determine whether an article supports or refutes a claim by using the three snippets with the highest stance scores. These scores help them to further divide an article into three categories: fully supportive, completely refuting, and a mixture of the former stances. Similar to [Nakashole and Mitchell, 2014], they capture the text subjectivity to detect high confidence stances. Some linguistic features are taken into account for this purpose: (1) *factive verbs* such as know, realize, etc. (2) *assertive verbs* such as think, believe, etc. (3) *mitigating words* such as about, apparently, etc. (4) *report verbs* such as admit, agree, etc. (5) *discourse markers* such as could, maybe, etc. (6) *subjective/bias words* such as accept, abuse, etc. These features are encoded into one-hot feature vectors.
- The *source credibility assessment* module measures the reliability of article's source. They obtain this assessment by using Web of Trust (WOT)[22]. In another study, [Nadeem et al., 2019] use the Media Bias/Fact Check website[23] to assess the credibility of media sources. They obtain the three labels *high or very high*, *low and low questionable*, and *mixed* for 2,500 news websites.

Finally, the truthfulness of a given claim is measured as a weighted sum of the stance scores provided by the article classifier and the WOT score. On a test set of 105 claims, the authors report an accuracy of 83.63% for the stance classifier and an accuracy of 85.25% for the task of predicting claim truthfulness.

3.4.2 Using a knowledge graph

Given a statement that is described as an RDF triple *(subject, predicate, object)*, e.g., *(Chicago, isCapitalOf, Illinois)*, [Shi and Weninger, 2016] first determine the type of the subject, in this occurrence *city*, and that of the object, in this example *state*. Then they collect the set T^+ of node pairs of the form (instance of type city, instance of type state) that are connected by any path in the knowledge graph. Similarly, they collect the set T^- of node pairs of the form (instance of type city, instance of type state) that are *not* connected by any path in the knowledge graph. The set of predicates that connect node pairs in T^+ and T^- is called *meta path*. Meta paths obtained from T^+ and T^- lead to the positive and negative training examples, respectively. They prune meta paths considered irrelevant by calculating the *information gain* of each path and retaining the top-k ones. Information gain is calculated with the following formula:

$$IG(X_j, y) = \sum_{x_{ij} \in X_j} \sum_{y_i \in y} p(x_{ij}|y_i) \log \frac{p(x_{ij}|y_i)}{p(x_{ij})p(y_i)}$$

[22] https://mywot.com/wiki/API
[23] https://mediabiasfactcheck.com/

where:

- X is the training dataset consisting of n training examples $x_1, x_2, \ldots, x_n$. Each training example x_i is an m-dimensional vector representing the meta path $(p_{i_1}, p_{i_2}, \ldots, p_{i_m})$. x_{ij} is the number of node pairs connected by the predicate p_{i_j};
- $X_j = (x_{1j}, x_{2j}, \ldots, x_{nj})$ is the j-th column in X;
- $y_i \in \{0, 1\}$ is the label of the ith training example;
- y is the set of all labels;
- $p(x_{ij}) = \frac{x_{ij}}{\sum_{i,j} x_{ij}}$ is the relative importance of x_{ij}, the number of pairs connected by predicate p_{i_j}, among all the elements of the X matrix;
- $p(x_{ij}|y_i) = \frac{x_{ij}}{\sum_{k \in \{1,2,\ldots,n\}:y_k=y_i} x_{kj}}$ is the relative importance of x_{ij} among the elements in the column X_j having the label y_i
- $p(y_i) = \frac{\sum_{k \in \{1,2,\ldots,n\}:y_k=y_i} 1}{n}$ is the frequency of the label y_i within the training set.

Intuitively, the information gain reflects the relevance of a feature (a column of X) with respect to the target y. Removing irrelevant features result a better training set.

These top-k meta paths are used to train a binary logistic regression and to explain the predicted link between subject and object of the given statement.

Thus, in this work, truthfulness is assessed with the help of a **link prediction** approach, which intuitively states how likely the given triple is, by comparing with how frequent (likely) similar triples hold for similar (subject, object) pairs. This approach requires that the subject and object be typed; further, if one or both had several types, some extension to this method would be needed.

The authors present experiments carried on two large knowledge bases DBpedia [Lehmann et al., 2015] and SemMedDB [Kilicoglu et al., 2012]. When comparing against six other link prediction models, one fact checking model and an association rule mining model, they report the highest area-under-curve (AUC) on 5 out of 7 link-prediction tasks.

This work also provides human-interpretable, intuitive explanations of their prediction, by showing the most probable paths in the knowledge graph that connect the type of subject and object. A sample explanation when checking the above sample triple is: "a *US city* is likely to be the capital of a *US state* if an state agency with jurisdiction in that state has its headquarters in that city".

[Ahmadi et al., 2019] uses rules and evidence from knowledge graph (KG) in order to explain fact-checking outputs. A KG contains facts as RDF triples of the first $p(s, o)$ where p is a predicate connecting a subject s and an object o. For example, the claim "William Durant was the founder of Chevrolet" is true because of the two triples $keyPerson(Durant, Chevrolet)$ and $foundedBy(Durant, Chevrolet)$. The following claim "Elon Musk was the founder of Chevrolet" is false because of these three triples $foundingYear(Chevrolet, a)$, $birthYear(Musk, b)$, $greater(b, a)$. In a nutshell, the authors mine rules from the KG; to fact-check a claim represented as an RDF triple, they collect the evidence under the form of facts from the KG and

Web documents. Finally, they apply an answer set programming technique to determine the truthfulness of the claim. Below we give more details on these steps.

They adopt `RUDIK` [Ortona et al., 2017] to generate rules for each predicate in KG. A rule r is denoted $r : h(x, y) \leftarrow B_1(z_1, z_2) \wedge B_2(z_3, z_4) \wedge B_n(z_{2n-1}, z_2 n)$ where h is the head of the rule and all B_i triples form its body. The head of a rule is considered true (correct) when all triples in its body are true. An example is *spouse(a, b)* $\leftarrow$ *child(a, c)* $\wedge$ *child(b, c)*. This rule reads: *"a and b have the same child c, thusa and b participate in a spouse relation"*. They also add a rule to ensure that a claim and its contradiction cannot be true at the same time, and another rule to make sure that a claim can have only one object value.

Given a claim $p(s, o)$ and a rule r, they generate evidence for the claim by substituting x, y to all triples B_i ($i \in \{1, 2, \ldots, n\}$), then collect triples that satisfy r in KG.

The authors observe that one can not always rely solely on KG facts, since it may be incomplete. For example if we have a rule $r : p(x, y) \leftarrow A(x, z) \wedge B(y, z)$ and only $A(x, z)$ is found from KG then we need to find the fact $B(y, z)$ from other sources. To solve this problem, they apply `CredEye` [Popat et al., 2016] to harvest the missing facts from Web documents. Each fact has a confidence score, indicating whether the fact is true. They collect facts that have confidence scores greater than 0.5.

Given a claim c, a set of rules and a set of evidence, they apply probabilistic answer set programming [Lee and Wang, 2016] to output a subset of the rules R and a subset of the evidence E. By substituting triples from E into the bodies of rules from R, we could derive either c or the negation of c in the head of one rule r^* from R to conclude the claim is true or false respectively. They use r^* as the explanation for the fact-check result of c. If R and E are empty sets, they could not fact-check c.

They evaluate their system's performance on 4 predicates for a total of 2,400 claims. Compared with three baselines, their system have significant higher F1-scores for three predicates (0.88, 0.83, and 0.87).

3.4.3 Using linguistic features

[Nakashole and Mitchell, 2014] aim to determine the truthfulness of a fact such as "Einstein died in Princeton" by analyzing the presence of **language's level of objectivity**.

They use crowdsourcing to analyze the correlation between the objectivity of language and the trustworthiness of news articles. For each article, they ask workers to label it as *subjective* / *objective* and *trustworthy* / *untrustworthy*. An example of objective sentence is "Theories allege that Obama's published birth certificate is a forgery, that his actual birthplace is not Hawaii but Kenya". An example of subjective sentence is "Well, I think Obama was born in Kenya because his grandma who lives in Kenya said he was born there.". On a dataset of 420 articles, they report that 74% of the untrustworthy articles are also found subjective. On the other hand, 64% of trustworthy articles were labeled as objective. Thus, they form a hypothesis *objective text is more trustworthy than subjective one*.

To train an objectivity detector, they crowdsource a labeled dataset and reuse another dataset from prior work on subjectivity detection [Pang and Lee, 2004]. The final dataset consists of 4,600 documents. They train a binary classifier with five sets of features: (1) subjectivity lexicon of *strong* and *weak subjective* words, (2) sentiment lexicon of *positive* and *negative*

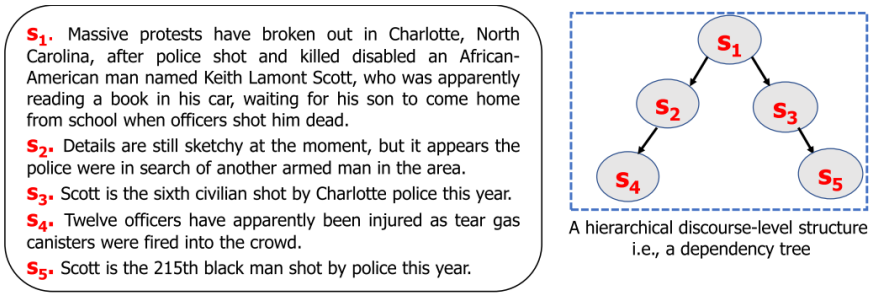

Figure 3.4: Discourse-level hierarchical representation of a document using dependency tree parsing [Karimi and Tang, 2019]

words, (3) Wikipedia-derived *bias* lexicon, (4) part-of-speech, (5) frequent bi-grams. They report a precision of 78.14%. The probability that the learned model assigns to the objective label is call *objectivity* of a document. The feature sets (1) and (2) are also adopted by [Nadeem et al., 2019] in a similar effort to analyze the language used in relevant documents of a claim. A new feature set *Wiki-bias lexicon*, bias cues and controversial words such as abortion and execute, are also introduced by [Nadeem et al., 2019].

A fact candidate is a piece of evidence connected to a claim. It can appear in one or more sources, where a source is a collection of documents such as the set of articles in a newspaper, or Wikipedia. A fact candidate is represented as a RDF triple $(subject, verbal_phrase, object)$.

Using the objectivity of a document, they compute the two following scores for a fact candidate:

- *objectivity score* to ensure that a fact candidate mentioned in 10 sources with 0.9 objectivity should be given more faith than a fact candidate stated in 1 source of 0.9 objectivity.
- *co-mention score* to gives "boost" to the fact candidate that is co-mentioned in the same source(s) with another fact candidate that has high objectivity score.

Finally they compute a *believability score* for each fact candidate which is a weighted sum of the two previous mentioned scores. The believability score is used to determine whether a fact candidate is true or false. High accuracy scores are reported on three different datasets:

- *KB dataset* provided by [Carlson et al., 2010] consisting of 190 fact candidates about company acquisitions, book authors, movie directors, and athlete teams.
- *Wikipedia dataset* consisting of 54 fact candidates about US politicians. They created this dataset manually.
- *General Knowledge Quiz dataset* consisting of 18 fact candidates collected from a general knowledge quiz[24].

[Karimi and Tang, 2019] develop a new approach to detect fake news based on the observation of **hierarchical structure (e.g., a tree data structure) of discourse units** (e.g., sentences) could provide important insight of document's truthfulness.

Their system consists of three components: *Discourse-level Hierarchical Structure Learning*, *Document-level Structural Representation*, and *Fake News Classification*.

[24] http://www.indiabix.com/general-knowledge/

- The first component learns the semantic dependencies between discourse units. These dependencies are illustrated in Figure 3.4. The output of this task is a tree where each node is a sentence of the document and the child node depends semantically on its parent node. They use Bi-LSTM (Section 2.2.3) to represent discourse units. Then a matrix A is constructed where $A(m, n)$ denotes the probability that sentence m is the parent sentence of sentence n. This matrix also helps computing the probability of being root node for each sentence. Finally, they build a dependency tree of all sentences using a greedy algorithm.
- The second component adopts the approach of [Liu and Lapata, 2018] to compute a structural representation of the document, as follows. Each sentence of the document is represented as the concatenation of its parent/child nodes' vector representation in the dependency tree and the vector representation of the sentence itself. The representation of the document is the average of its sentences' representations.
- The last component is a binary document classifier into fake, respectively, non fake, applied on the document representation built by the previous component.

On a dataset of 3,360 fake and 3,360 real documents, they obtained an accuracy of 82.19% on the test set. Furthermore, they discover that the degree of coherence of real documents is higher than those of fake ones. This is determined using three properties calculated from the discourse-level hierarchical structure: the number of leaf nodes, the parent-child distance, and the *preorder difference* which is a normalized positional difference between the preorder traversal of a document's discourse dependency tree and its original sentential sequential order.

3.4.4 Using user input

[Lim et al., 2017] propose **iFact**, an approach to verify claims' credibility which incorporates users' inputs. This approach collects evidence from web search, then assign to a claim the probability that it is *credible*, *not credible*, or *inconclusive*. Observing that a claim's credibility could be dependent on the credibilities of others, they detect the dependencies among claims and adjust the aforementioned probabilities with first order logic rules. Users can review and decide the credibility of claims with an interactive GUI (graphical user interface).

They extract claims from tweets using their ClaimFinder framework [Lim et al., 2016]. Each claim is represented in tuple format *(Subject, Predicate)* where *Subject* is a set of nouns and proper names, and *Predicate* is a set of verbs. For example, from the tweet *"Unconfirmed reports suggest flight MS804 landed. Not sure how true..."*, the extracted tuple is *(*{*flight ms804*}*,* {*land*}*)*.

The web search is performed with the union of words from *Subject* and *Predicate*. In the previous example, the search query would be *"flight ms804 land"*. Each web search result (WSR) consists of a source URL, its text title and snippet (the summary text of each URL). They extract the following features for each WSR:

1. **Results** is the number of results returned from the search engine.
2. **Doubt** is the fraction of top-k WSRs containing a question mark or some doubt words. They collect doubt words manually from a set of rumours from `snopes.com` and `truthorfiction.com`. Some examples are "gossip, incorrect, mislead, scam, etc.".

3. **Reputable** is the fraction of top-k WSRs coming from reputable sources. They consider a source reputable if it comes from a pre-defined list of well-known sites such as reuteurs.com, bbc.co.uk. The **trust score**[25] is also taken into account to determine the reputation of a source.

4. **Reputable doubt** is the intersection of the **Doubt** and **Reputable** features.

5. **Trust score** is the average trust score of sources from top-k WSRs.

6. **Doubt score** is the average trust score of sources from top-k WSRs express doubt.

After manually labeling claims as credible, not credible, or inconclusive, they train a supervised learning model with the above features to obtain the probabilities of each class. The obtained precision, recall, and F1-score outperform the scores of the same model training with tweet-based features [Castillo et al., 2011] that focus on tweet's linguistic features and Twitter user's information.

They consider two types of dependencies between two claims that refer to a similar subject: *direct dependency* if the two have the same credibility, and *inverse dependency* if one claim is credible and the other is not credible. For instance, a direct dependency occurs when the two claims' predicates are synonyms. An inverse dependency occurs when the two claims' predicates are antonyms.

The similarity of two subjects is measured by the cosine similarity between their vector representations, using Doc2Vec.

Using Probabilistic Soft Logic [Bach et al., 2017], they formulate first order logic rules to assign the same class to the two claims with a direct dependency, and assign opposite classes to the two claims with an inverse dependency.

The iFact system takes into account inputs from users, by providing a GUI that allow users to decide whether a WSR is relevant to a given claim. The GUI also lets users change the type of dependency among the claims. Finally, iFact recomputes the credibility of claims with the new inputs from users.

[Nguyen et al., 2019] propose a method to verify the credibility of claims by making an efficient use of user inputs. To explain their approach, we recall a set of notations they introduce. D is a set of documents, where a document can be a tweet , a news item, or a forum posting, etc.; S is a set of data sources, where a source can be, for example, a web site, or a news provider; C is a set of claims; P is probabilistic model, where $P(c = 1)$ refers to the probability that a given claim $c \in C$ is *credible*. They define a *probabilistic fact database* $Q = (S, D, C, P)$.

They perform the following steps to validate claims:

1. selecting a claim c from C using an *user guidance approach* that will be described below,

2. getting user input to confirm the credibility of the claim (label the claim as 1 for "credible" or 0 for "non-credible"),

3. updating the probabilistic credibility model P,

4. deciding whether the claim is credible, then go back to step 1.

[25] https://www.mywot.com/

This process is iterated until an *effort budget* (the maximum number of iterations) or a *validation goal* such as a certain threshold of P's precision is matched.

They perform step 3 using iCRF, an implementation they propose for the CRF [Lafferty et al., 2001] model, which does not need to re-compute P and the model parameters after each iteration. iCRF takes D and S feature vectors as input to compute P. The construction of these feature vectors is not clearly explained in [Nguyen et al., 2019][26].

The core of their approach is the *user guidance* that tries to reduce the *uncertainty* of Q by selecting the "best" claim c for the user to validate. They present three methods for selecting the next claim for which to solicit user guidance:

1. The **information-driven approach** starts by quantifying the uncertainty of Q and approximating it in linear time as follows:

   $H_C(Q) = -\sum_{c \in C}(Pr(c)\log Pr(c) + (1 - Pr(c))\log(1 - Pr(c)))$ where $Pr(c)$ is the probability that the claim c is credible. This probability is obtained from iCRF for unlabeled claims, or from user input for labeled claims. This can be seen as a form of active learning.

   The claim to select is the one that maximizes its information gain. This is computed as: $IG_C(c) = H_C(Q) - Pr(c)H_C(Q^+) - (1 - Pr(c))H_C(Q^-)$ where Q^+ and Q^- are the subsets of Q inferred by iCRF (based on user input) as holding the credible and non-credible claims, respectively.

2. The **source-driven** approach aggregates the credible claims from a source to assess that source's credibility. The likelihood that a source s is trustworthy, $Pr(s)$, is computed as the fraction of its claims that are considered credible.

   The uncertainty of source trustworthiness is measured by

   $$H_S(Q) = -\sum_{s \in S}(Pr(s)\log Pr(s) + (1 - Pr(s))\log(1 - Pr(s)))$$

   Similarly to the information-driven approach, we select the claim maximizing information gain:

   $$IG_S(c) = H_S(Q) - Pr(s)H_S(Q^+) - (1 - Pr(s))H_S(Q^-)$$

3. A **hybrid** approach combines the information- and source-driven methods. In each iteration, they compute a score from the ratio of untrustworthy sources and the credible probability of claims to decide whether to choose the information-driven or source-driven for the next iteration.

To evaluate their method, they simulate the user inputs with labeled claims from the Snopes dataset [Popat et al., 2017]. Their approach achieves a precision > 0.9 after using only 31% of the claims. Other baselines require at least 67% of the claims to obtain the same precision.

[26] *"We abstract from the specific nature of these features, but take into account that the trustworthiness of a source and the language quality of a document have a strong influence on the credibility of the claims."*

3.5 Fact checking challenges

A number of challenges has been organized to tackle the tasks of stance detection (Section 3.5.1), fact extraction and verification (Section 3.5.2), and check worthiness (Section 3.5.3).

3.5.1 Fake news challenge

The **Fake News Challenge** [Pomerleau and Rao, 2017] focuses on the *stance detection* task, which consists of classifying a news article among four classes: "agree", "disagree", "discuss", and "unrelated" with respect to a given headline (or claim). The winning solution [Baird et al., 2017] is an ensemble of gradient-boosted decision trees (GBDT) model and convolutional neural networks (CNNs). The authors published their source code on GitHub[27].

Their GBDT model was trained on the following features: (1) the number *overlapping words* between the headline and the article body text; (2) the similarities measured between their word counts, 2-grams and 3-grams; and (3) similarities measured after transforming these counts with TF-IDF [Jones, 1972] weighting and Singular Value Decomposition (SVD) [Klema and Laub, 1980]. Features based on word2vec and sentiment analysis are also said to have been used, but the details are not mentioned in their blog post.

For the CNN model, firstly they represent the headline and article body at word level using pre-trained word2vec vectors. For each training example, the representations of the headline and the body are concatenated to form a vector. These vectors are inputs of CNN layers, regularized using Dropout [Srivastava et al., 2014]. The output is then sent to a multilayer perceptron (MLP) [Hornik et al., 1989], which determines the relationship between the news article body and the headline.

3.5.2 Fact Extraction and VERification

The **FEVER** challenge [Thorne et al., 2018] asks participant systems to verify an input claim (a sentence) against a corpus of 5 million Wikipedia documents. This challenge provides a large scale benchmark for developing new document retrieval systems. In the fact-checking context, it helps to retrieve factual data from text. These systems are also asked to provide the evidential sentences from the corpus in order to label the claim as "supports", "refutes", or "not enough info". Each system usually consists of three sub-systems to solve three tasks:

1. document retrieval to select the relevant documents with respect to the given claim,
2. sentence selection to select the evidential sentences,
3. claim verification to label the input claim.

The winners of this challenge are [Nie et al., 2019]. They introduced a neural network architecture called Neural Semantic Matching Network to tackle the three sub-tasks as a similar textual semantic matching problem. This network outputs a relatedness score between two pieces of text. For the first sub-task, they compute the semantic relatedness between the input claim and the representation of the concatenation of the title and the first sentence of a document. They

[27] https://github.com/Cisco-Talos/fnc-1/

apply keyword matching to reduce the amount of candidate documents. Similarly, the sentence selection sub-task is handled by computing the relatedness score between the claim and every sentence from the documents retrieved from the first sub-task. Finally, they use the relatedness scores of the set of evidential sentences and the claim, and some additional features (from WordNet and the previous two sub-tasks) to output the probabilities for each of three labels.

3.5.3 Check worthiness

The **CLEF-2018 CheckThat! Task 1: Check-Worthiness** [Barrón-Cedeño et al., 2018] asks to predict if a given claim from a political debate should be prioritized for fact checking.

[Zuo et al., 2018] was ranked first on this task. Their system is a multilayer perceptron (MLP), based on the following features characterizing each input sentence:

- *sentence embedding*: the average of the embeddings of the words in the sentence
- *stylometric features* such as; the number of words; the number of words in past, present, and future tenses; the number of negations; the number of words within each clause and phrase generated by the constituency parse tree of the input sentence;
- *semantic features* consisting of the number of named entities; the number of named entities of the type PERSON.
- *sentiment features* using Connotation WordNet [Kang et al., 2014] and sentiment score [Pang et al., 2002]. They also capture the subjectivity of words using the lexicon from [Wilson et al., 2005].
- *metadata features* to indicate if the speaker's opponent is mentioned, the moderator is the speaker, or the sentence is followed immediately by a sentence from the moderator of the debate.
- *discourse features* such as the relative position of a sentence within its segment, where a segment is a set of consecutive sentences from the same speaker.

3.6 Automated end-to-end fact checking systems

ClaimBuster [Hassan et al., 2017] claimed to be the first end-to-end fact checking system. Their system architecture is illustrated in Figure 3.5. It performs fact-checking automatically using the following components:

1. The *claim monitor* continuously retrieves texts from different sources such as broadcast media, social media, and websites in order to discover factual claims.
2. The *claim spotter* gives a score between 0.0 and 1.0 to a given sentence, to indicate its check-worthiness. They report recall and precision of this component of 74% and 79% respectively.
3. The *claim matcher* searches the best matching fact-checks curated from various fact checking websites to a given claim. To determine the ranking order, they combine token

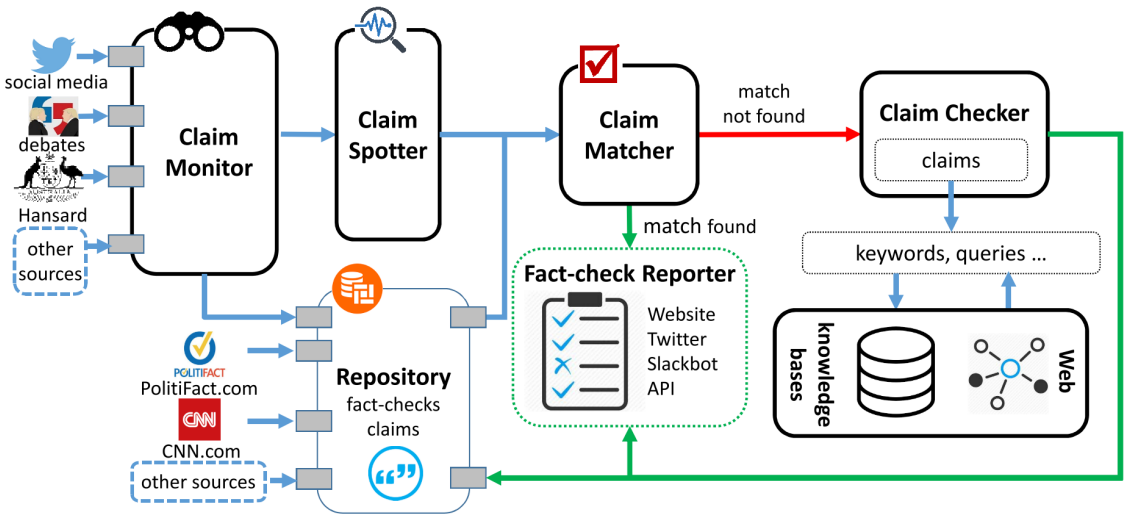

Figure 3.5: ClaimBuster's system architecture [Hassan et al., 2017]

similarity from an Elastic search[28] server and semantic similarity from Semilar toolkit [Rus et al., 2013].

4. The *claim checker* uses a question generation tool [Heilman and Smith, 2009] to generate questions from the claim. Then, they pick the most suitable ones to send to the question answering engine Wolfram Alpha[29] and to the Google search engine.

5. The *fact-check reporter* delivers outputs of the previous components to the end-users. Their Twitter account have been tweeting 13K check-worthy factual claims from politicians and organizations. They also created a public ClaimBuster API to help developers develop fact checking applications.

[Nadeem et al., 2019] present the end-to-end FAKTA fact checking system[30] as the integration of various components: document retrieval, stance detection, evidence extraction, and linguistic analysis.

The stance detection is handled by the state-of-the-art system on the Fake News Challenge [Pomerleau and Rao, 2017] developed by [Xu et al., 2019]. The model outputs a stance score for a document with respect to a claim. They also improve the model by using an *adversarial domain adaptation* technique which helps it overcome the limited size of labeled data when training through different domains.

The rationale of their model are explained by the stance score for each sentence in the relevant documents to the given claim. These sentences are highlighted and color coded in a web interface in order to help end-user spot these important information more easily. The frequency of each lexicon type (sentiment cues [Riloff and Wiebe, 2003], subjectivity lexicon [Liu et al., 2005], and Wiki-bias lexicon [Recasens et al., 2013]) is taken into account in order to provide lexicon-specific word clouds and lexicon-specific scores as a radar chart for the end-user. An illustration of the FAKTA system is presented in Figure 3.6.

[28] https://github.com/elastic/elasticsearch
[29] http://products.wolframalpha.com/api
[30] http://fakta.mit.edu/

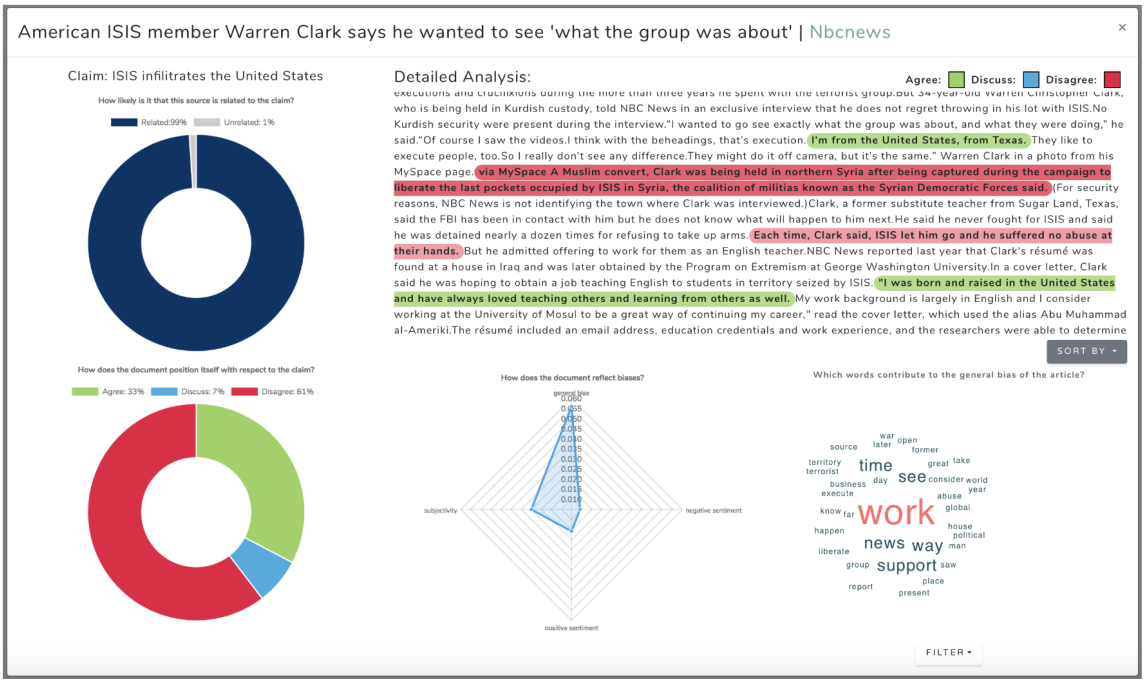

Figure 3.6: A document retrieved for the claim “ISIS infiltrates the United States” [Nadeem et al., 2019]

Chequeado[31] is a system similar to ClaimBuster [Hassan et al., 2017]. As of 2018, the program could monitor the presidential speeches and articles from 30 Argentine media outlets to detect check-worthy claims [Graves, 2018]. They work closely with fact-checkers to improve their algorithm. They are developing automated verification methods for a given claim, based on previous fact-checks and official statistics. Unfortunately, details of their methods are currently not available.

FullFact’s Live platform[32] detects claims in TV’s subtitles and then propose the relevant articles. In case the detected claim has not been fact-checked, the system could create fact-checks on the spot using reliable data.

3.7 Conclusion

The literature provides us many approaches to leverage external data, e.g., Wikipedia, knowledge graph, etc. and linguistic features to tackle the fact-checking task. The availability of datasets and benchmarks let researchers to try their new ideas more easily. We also see some end-to-end systems indicating that computational fact-checking is feasible.

There are also some limitations of the existing systems. Firstly most of these works focus only in English language. In order to apply these existing approaches to a new language, we have to build new tools to work with the new language. We also notice the lack of attention on spreadsheet/tabular data which is really helpful in practice for fact-checking statistic claims.

[31] https://chequeado.com/
[32] https://fullfact.org/

In our work, we develop our algorithms and software from scratch as we focus on spreadsheet data and French language.

Chapter 4

Extracting linked data from statistic spreadsheets

4.1 Introduction

The tremendous value of Big Data has been noticed of late also by the media. While data of some form is a natural ingredient of all reporting, the increasing volumes of available digital data as well as its increasing complexity lead to a qualitative jump, where technical skills for working with data are stringently needed in journalism teams.

A class of data sources of particularly high value is that comprised into *government statistics*. Such data has been painstakingly collected by state administrations which sometimes have very significant human and technology means at their disposal. This makes the data oftentimes of high quality. Another important quality of such data is that it logically falls into the open data category, since it is paid for with taxpayer money.

Such high-quality statistic data is a very good background information to be used when trying to assess the truthfulness of a claim. A fact-checking scenario typically has several steps: first, a *claim* is made. Second, journalists or other fact-checkers look up *reference sources* providing relevant statistics for the assessment of the claim; Third, some *post-processing* may be applied to compute from the reference source values, the value corresponding to the claim. This post-processing may involve elaborate steps.

In this chapter, we focus on the extraction of reference sources. Ideally, these are available to fact-checkers in a format and organized according to a model that they understand well. For instance, in [Wu et al., 2014], unemployment data is assumed available in a temporal database relation. In reality, however, most fact-checking actors only have access to the data through *publication* by the institute. This raises the question of the format and organization of the data. While the W3C's best practices argue for the usage of RDF in publishing open data [The World Wide Web Consortium (W3C), 2014], in practice open data may be published in Excel, HTML or PDF; in the latter formats, statistic data is typically shown as tables or charts. In the particular case of the **INSEE**[1], the national French social and economic statistics institute, while searching for some hot topics in the current French political debate (youth unemployment, immigrant population etc.) we found many very useful datasets in Excel and/or

[1] https://insee.fr

HTML, sometimes accompanied by a PDF report explaining and commenting the data, but we did not find the data in a structured, machine-readable format.

Motivated by this limitation, we propose an approach for extracting Linked Open Data from INSEE Excel tables. The two main contributions of our work are:

1. a conceptual data model (and concrete RDF implementation) for statistic data such as published by INSEE and other similar institutions,
2. and an extraction algorithm which populates this model based on about $11,000$ pages including $20,743$ Excel files published by INSEE.

Below, we describe the data sources we work with and the model for the extracted data (Section 4.2) and our extraction method (Section 4.3). The RDF vocabulary we use in the extracted data is outlined in Section 4.4. Finally we present an evaluation of our extraction process (Section 4.5) as well as the implementation details (Section 4.6). We mention the related works (Section 4.7) before concluding (Section 4.8).

We were aware of some APIs from INSEE. The Sirene API [2] focuses on enterprise' information. The Nomenclatures API [3] was available from July 2019 and we worked on this project in 2016 and 2017. INSEE also provides data access in RDF format [4]. Time series data are provided via SDMX API [5]. All of the aforementioned APIs provide sub-sets of INSEE data grouped by theme such as enterprise, time series, etc. Our work focuses on extracting all INSEE data and store them in a unified data format for further access.

The content of this chapter was presented in a publication in the International Workshop on Semantic Big Data 2017 [Cao et al., 2017].

4.2 Reference statistic data

We present the input INSEE statistics (Section 4.2.1), then describe our conceptual model for this data (Section 4.2.2).

4.2.1 INSEE data sources

INSEE publishes a variety of datasets in many different formats. In particular, we performed a crawl of the INSEE web site under the categories *Data/Key figures* (*Données/Chiffres-clés*), *Data/Detailed figures* (*Données/Chiffres-détaillés*), *Data/Databases* (*Données/Base de données*), and *Publications/Wide-audience publications* (*Publications/Publications grand public*) , which has lead to about 11,000 web pages including 20,743 Excel files (with total size of 36GB) and 20,116 HTML tables. In this work, we focused on the set of Excel files; we believe our techniques could be quite easily adapted to HTML tables in the future. Also, while we targeted

[2] https://api.insee.fr/catalogue/site/themes/wso2/subthemes/insee/pages/item-info.jag?name=Sirene&version=V3&provider=insee

[3] https://api.insee.fr/catalogue/site/themes/wso2/subthemes/insee/pages/item-info.jag?name=Nomenclatures&version=v1&provider=insee

[4] http://rdf.insee.fr/index.html

[5] https://www.insee.fr/en/information/2868055

INSEE in this work, we found similar-structure files published in other Open Data government servers in the UK[6] and the US[7].

l \ c	1	2	3	4	5	6	7	8	9	10
1	The data reflects children born alive in 2015...									
2										
3			Mother's age at the time of the birth							
4			Age below 30			Age above 31				
5	**Region**	**Department**	**16-20**	**21-25**	**26-30**	**31-35**	**36-40**	**41-45**	**46-50**	
6	Île-de-France	Essonne	215	1230	5643	4320	3120	1514	673	
7		Val-de-Marne	175	987	4325	3156	2989	1740	566	
8		...	...	...	...	...	...	...	...	
9	Rhône-Alpes	Ain	76	1103	3677	2897	1976	1464		
10		Ardèche	45	954	2865	2761	1752	1653	523	
11		...	...	...	...	...	...	...	...	
...	...	...	...	...	...	...	...	...	...	

Figure 4.1: Outline of a sample statistics table in an INSEE dataset.

We view each spreadsheet file as a collection of *data sheets* $D_1, D_2, \ldots$ Each data sheet D has a *title* $D.t$ and optionally an *outline* $D.o$. The title and outline are given by the statisticians producing the spreadsheets in order to facilitate their understanding by human users, such as the media and general public. The title is a short nominal phrase stating what D contains, e.g., "Average mothers' age at the birth of their child per department in 2015". The outline, when present, is a short paragraph which may be found on a Web page from where the file enclosing D can be downloaded. The outline extends the title, for instance to state that the mothers' ages are rounded to the closest smaller integers, that only the births of live (not still-born) children are accounted for etc.

In a majority of cases (about two thirds in our estimation), each data sheet D comprises *one* statistics table, which is typically a *two-dimensional aggregate table*. Data is *aggregated* since statistics institutes such as INSEE are concerned with building such global measures of society or economy phenomena, and the aggregate tends to be *bidimensional* since they are meant to be laid out in a format easy for humans to read. For instance, Figure 4.1 illustrates a data sheet for the birth statistic dataset mentioned above. In this example, the two dimensions are the mother's age interval, e.g., "16-20", "21-25", "26-30" etc. and her geographic area, specified as a department, e.g., "Essonne", "Val-de-Marne" etc. For a given age interval and region, the table includes the number of women which were first-time mothers in France in 2015, in that age interval and department. In Figure 4.1, we have included a top row and column on a gray background, in order to refer to the row and column numbers of the cells in the sheet. We will use the shorthand $D_{r,c}$ to denote the cell at row r and column c in D. We see that the table contains *data cells*, such as $D_{6,3}$, $D_{6,4}$, which hold data values, etc. and *header cells*, e.g., "Region", "Age below 30" etc., which (i) characterize (provide context for) the data values, and (ii) may occupy several cells in the spreadsheet, as is the case of the latter.

The basic kind of two-dimensional aggregate in a data sheet is (close to) the result of a group-by query, as illustrated in Figure 4.1. However, we also found cases where the data sheet contains a *data cube* [Gray et al., 2007], that is: the result of a group-by, enhanced with partial sums along

[6] http://www.ic.nhs.uk/catalogue/PUB08694/ifs-uk-2010-chap1-tab.xls

[7] http://www.eia.gov/totalenergy/data/monthly/query/mer_data_excel.asp?table=T02.01

some of the group-by dimensions. For instance, in the above example, the table may contain a row for every region labeled "Region total" which sums up the numbers for all departments of the region, for each age interval[8].

Dimension hierarchies are often present in the data. For instance, the dimension values "16-20", "21-25", "26-30" are shown in Figure 4.1 as subdivisions of "Age below 30", while the next four intervals may be presented as part of "Age above 31". The age dimension hierarchy is shown in magenta in Figure 4.1, while the geographical dimension hierarchy is shown in blue. In those cases, the lower-granularity dimension values appear in the table close *close to the finer-granularity dimension values which they aggregate*, as exemplified in Figure 4.1. *Cell fusion* is used in this case to give a visual cue of the finer-granularity dimension values corresponding to the lower-granularity dimension value which aggregates them.

A few more rules govern data organization in the spreadsheet:

1. $D_{1,1}$ (and in some cases, more cells on row 1, or the first few rows of D) contain the outline $D.o$, as illustrated in Figure 4.1; here, the outline is a long text, thus a spreadsheet editor would show it over several cells. One or several fully empty rows separate the outline from the topmost cells of the data table; in our example, this is the case of row 2. Observe that due to the way data is spatially laid out in the sheet, these topmost cells are related to the lowest granularity (top of the hierarchy) of one dimension of the aggregate. For instance, in Figure 4.1, this concerns the cell at line 3 and columns 3 to 9, containing **Mother's age at the time of the birth**.

2. Sometimes, the content of a header cell (that is, a value of an aggregation dimension) is not human-understandable, but it is the name of a variable or measure, consistently used in all INSEE publications to refer to a given concept. Exactly in these cases, the file containing D has a sheet immediately after D, where the variable (or measure) name is mapped to the natural-language description explaining its meaning. For example, we translate SEXE1_AGEPYR1018[9] into *Male from 18 to 24 years old* using the table below:

Variable	**Meaning**
SEXE	**Sex**
1	Male
2	Female
AGEPYR10	**Age group**
00	Less than 3 years old
03	3 to 5 years old
06	6 to 10 years old
11	11 to 17 years old
...	...
80	80 years and older

4.2.2 Conceptual data model

From the above description of the INSEE statistic datasets, we extract a conceptual data model shown in Figure 4.2. Entities are shown as boxes while attributes appear in oval shapes. We

[8] https://www.insee.fr/fr/statistiques/fichier/2383936/ARA_CD_2016_action_sociale_1-Population_selon_l_age.xls

[9] https://www.insee.fr/fr/statistiques/fichier/2045005/BTX_TD_POP1A_2013.zip

simplified the notation to depict relationships as directed arrows, labeled with the relationship name. *Data cells* and *header cells* each have an attribute indicating their value (content); by definition, any data or header cell has a non-empty content. Moreover, a data cell has one associated *location*, i.e., (row number, column number), while a header cell may occupy one or several (adjacent) locations. Each data cell has a closest header cell on its column, and another one on its line; these capture the dimension values corresponding to a data cell. For instance, the closest column header cell for the data cell at $D_{7,4}$ is $D_{5,4}$, while the closest column header cell is $D_{7,2}$. Further, the header cells are organized in an aggregation hierarchy materialized by the respective relation "aggregated by". For instance, $D_{5,4}$ is aggregated by $D_{4,3}$, which is aggregated by $D_{3,3}$. In each sheet, we identify the top dimension values as those which are not aggregated by any other values: in our example, the top dimension values appear in $D_{3,3}$, $D_{6,1}$ and $D_{9,1}$. Note that *such top dimension values are sometimes dimensions names*, e.g., **Region**, and other times *dimensions values*, e.g., Île-de-France. *Statistics in a sheet are not always organized exactly as predicted by the relational aggregate query models*, even if they are often close.

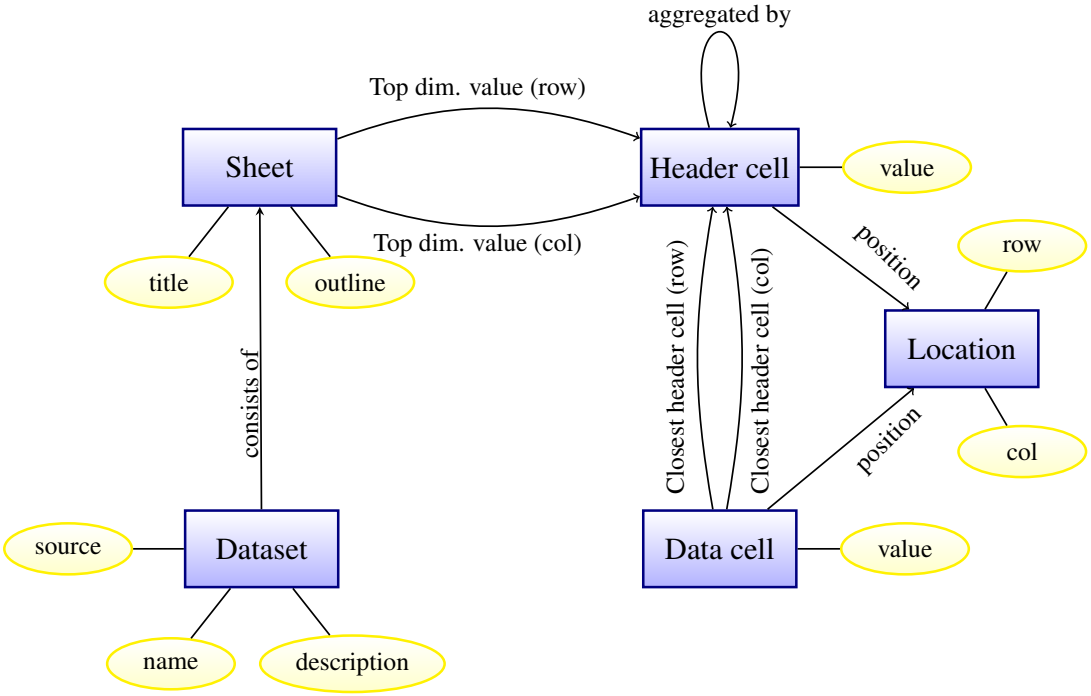

Figure 4.2: Conceptual data model.

We chose not to define precise types, e.g., population, economic growth, etc., for each spreadsheet because in the next chapter (Chapter 5) we will develop a search engine based on textual content in each dataset. The design of the conceptual data model allows us to query quickly necessary facts since it captures the hierarchy of headers and the data cell corresponding to a pair of headers. We also decided not to apply supervised machine learning because it requires labeled data which is time consuming to create. In our settings, the rule-based approach to extract data from spreadsheets is more appropriate.

Finally, note that our current data model does not account for data cubes, that is: we do not model the cases when some data cells are aggregations over the values of other data cells. Thus, our current extraction algorithm (described below) would extract data from such cubes as if they were result of a plain aggregation query. This clearly introduces some loss of information; we

are working to improve it in a future version.

4.3 Spreadsheet data extraction

We developed a tool for extracting data from spreadsheet files so as to populate an instance of our conceptual data model; it uses the Python library xlrd[10] for converting Excel file to a matrix of rows and columns. Below, we present the extraction procedure as a rule-based system aiming at framing the data cell range (Section 4.3.1) and extracting the headers (Section 4.3.2) around these data cells. Finally, we populate the data model with the extracted information (Section 4.3.3).

4.3.1 Data cell identification

For each sheet, the first task is to *identify the data cells*. In our corpus, these cells contain *numeric* information, that is: integers or real numbers, but also interval specifications such as "20+" (standing for "at least 20"), special *null* tokens such as "n.s." for "not specified". Our approach is to identify the *rectangular area containing the main table (of data cells)*, then build data cells out of the (row, column) locations found in this area. Note that the area may also contain empty locations, corresponding to the real-life situation when an information is not available or for some reason not reported, as is the case for $D_{9,9}$ in Figure 4.1. This process proceeds in several steps as illustrated in Figure 4.3.

l \ c	1	2	3	4	5	6	7	8	9	10
1	The data reflects children born alive in 2015...									
2										
3			Mother's age at the time of the birth							
4			Age below 30			Age above 31				
5	**Region**	**Department**	**16-20**	**21-25**	**26-30**	**31-35**	**36-40**	**41-45**	**46-50**	
6		Essonne	215	1230	5643	4320	3120	1514	673	
7	Île-de-France	Val-de-Marne	175	987	4325	3156	2989	1740	566	
8		...	...	...	...	...	...	...	...	
9		Ain	76	1103	3677	2897	1976	1464		
10	Rhône-Alpes	Ardèche	45	954	2865	2761	1752	1653	523	
11		...	...	...	...	...	...	...	...	
...	...	...	...	...	...	...	...	...	...	

Figure 4.3: Data cells extraction with leftmost data location (column 3, in brown), data table seed from row signatures (rows 6-10, in red), and additional data cells (row 9)

4.3.1.1 The leftmost data location

We identify the leftmost data location (ldl) on each row: this is the first location encountered from left to right which

1. follows at least one location containing text (such locations will be interpreted as being part of header cells),

[10] https://pypi.python.org/pypi/xlrd

2. and has a non-empty content that is compatible with a data cell (as specified above).

We store the column numbers of all such leftmost data locations in an array called ldl; for instance, $ldl[1] = -1$ (no data cell detected on row 1) and the same holds for rows 2 to 5, while $ldl[i] = 3$ for $i = 6, 7$ etc.

4.3.1.2 Row signature

For each row i where $ldl[i] \neq -1$, we compute a *row signature* $sig[i]$ which is counts the non-empty locations of various types found on the row. For instance, $sig[6] = \{string : 2, num : 7\}$ because the row has 2 string-valued cells ("Île-de-France" and "Essonne") and 7 numerical cells. For each signature thus obtained, we count the number of rows (with data cells) having exactly that signature; the largest row set is termed the *data table seed*. Note that the rows in the data table seed are not necessarily contiguous. This reflects the observation that slightly different-structured rows are sometimes found within the data table; for instance, $D_{9,9}$ is empty, therefore row 9 is not part of the seed, which in this example, consists of the rows $\{6, 7, 8, 10\}$ and possibly other rows (not shown) below.

4.3.1.3 Collect additional data cells

We "grow" the data table seed it by adding *every row at distance 1 or at most 2 above and below the data table seed, and which contains at least a data cell*. We do this repeatedly until no such rows can be found. In our example, this process adds row 9, despite the fact that it does not have the same signature as the others. We repeat this step in an inflationary fashion until no row adjacent to the current data table qualifies. The goal of this stage is to make sure we do not miss interesting data cells by being overly strict w.r.t. the structure of the rows containing data cells.

At this point, all the data cells are known, and their values and coordinates can be extracted.

4.3.2 Identification and extraction of header cells

We first look for those organized in *rows*, e.g., rows 3, 4 and 5 in our example. The process is illustrated in Figure 4.4.

4.3.2.1 The horizontal border

We attempt to exploit the visual layout of the file, by looking for a horizontal *border* which separates the first header row of the (usually empty) row above it.

- When such a border exists (which represents a majority but not all the cases), it is a good indicator that header cells are to be found *in the rows between the border and going down until (and before) the first data row*. (Note that the data rows are already known by now.) In our example, this corresponds to the rectangular area whose corners are $D_{3,1}$ and $D_{4,9}$.
- When the border does not exist, we inspect the rows starting immediately above the first data row and moving up, as follows. In each row i, we examine the locations starting at

l \ c	1	2	3	4	5	6	7	8	9	10
1	The data reflects children born alive in 2015...									
2										
3			Mother's age at the time of the birth							
4			Age below 30			Age above 31				
5	**Region**	**Department**	**16-20**	**21-25**	**26-30**	**31-35**	**36-40**	**41-45**	**46-50**	
6	Île-de-France	Essonne	215	1230	5643	4320	3120	1514	673	
7		Val-de-Marne	175	987	4325	3156	2989	1740	566	
8		...	...	...	...	...	...	...	...	
9	Rhône-Alpes	Ain	76	1103	3677	2897	1976	1464		
10		Ardèche	45	954	2865	2761	1752	1653	523	
11		...	...	...	...	...	...	...	...	
...	...	...	...	...	...	...	...	...	...	

Figure 4.4: Header cells extraction with rows to extract header cells (rows 3-4, in brown) and columns to extract header cells (columns 1-2, in orange)

the column $ldl[i]$ (if this is greater than 0). If the row doesn't contain empty locations, we consider that it is part of the header area. Intuitively, this is because no data cell can lack a value along one of the dimensions, and header cells carry exactly dimension values. An empty header cell would correspond precisely to such a missing dimension value.

4.3.2.2 Cell borders

Once the header area is known, we look for conceptual header cells. The task is complicated by the fact that such a cell may be spread over several locations: typically, a short text is spread from the top to the bottom over locations situated one on top of the other. This can also happen between locations in the same row (not in the same column), which are fused: this is the case of $D_{3,3}$ to $D_{3,9}$ in our example. The locations holding a single dimension value are sometimes (but not always) fused in the spreadsheet; sometimes, empty locations are enclosed within the same border as some locations holding text, to obtain a visually pleasing aspect where the dimension value "stands out well". We detect such groups by *tracking visible cell borders*, and create one conceptual header cell out of each area border-enclosed area.

4.3.2.3 Collect header cells

We *identify and extract header cells organized in columns*, e.g., columns 1 and 2 in our example. This is done by traversing all the data rows in the data area, and collecting from each row i, the cells that are at the left of the leftmost data cell $ldl[i]$.

4.3.3 Populating the data model

We populate the instance of the data model with the remaining relationships we need. We connect: each data and header cell to its respective locations; each data cell with its closest header cell on the same column, and with its closest header cell on the same row; each row (respectively, column) header cell to its closest header cell above (respectively, at left) through the relation aggregated by.

```
<http://inseeXtr.excel:File0>
      rdf:type <http://inseeXtr.excel:Dataset>;
      inseeXtr:source ”http://insee.fr/.../Age_Mère_....xls”;
      inseeXtr:name ”Age_Mère_....xls”;
      inseeXtr:description ”The data reflects children...” .
<http://inseeXtr.excel:Sheet0>
      rdf:type <http://inseeXtr.excel:Sheet>;
      inseeXtr:title ”sheet_title”;
      inseeXtr:belongsTo <http://inseeXtr.excel:File0> .
<http://inseeXtr.excel:HeaderCellY0> inseeXtr:value ”x1”;
      rdf:type <http://inseeXtr.excel:HeaderCellY>;
      inseeXtr:YHierarchy <http://inseeXtr.excel:Sheet0> .
<http://inseeXtr.excel:HeaderCellX2> inseeXtr:value ”a1”;
      rdf:type <http://inseeXtr.excel:HeaderCellX>;
      inseeXtr:aggregatedBy <http://inseeXtr.excel:HeaderCellX3>;
      inseeXtr:XHierarchy <http://inseeXtr.excel:Sheet0> .
<http://inseeXtr.excel:DataCell0> inseeXtr:value “1”;
      rdf:type <http://inseeXtr.excel:DataCell>;
      inseeXtr:location <http://inseeXtr.excel:Location0>;
      inseeXtr:closestXCell <http://inseeXtr.excel:HeaderCellX2>;
      inseeXtr:closestYCell <http://inseeXtr.excel:HeaderCellY0>.
<http://inseeXtr.excel:Location0>
      rdf:type <http://inseeXtr.excel:DataCell>;
      inseeXtr:Row “2”;
      inseeXtr:Col “3” .
```

Figure 4.5: Sample extracted RDF triples.

4.4 Linked data vocabulary

To represent the extracted data in RDF, we created an RDF class for each entity, and assigned an URI to each entity instance; the properties and their typing follow directly from the entity attributes and relationships shown in Figure 4.2.

Some sample triples resulting from the extraction are shown in Figure 4.5. We use the namespace `inseeXtr.excel:` for classes and properties we introduce in our extraction from INSEE Excel tables, and the namespace rdf: to denote the standard namespace associated to the W3C's type property. The snippet in Figure 4.5 corresponds to one Excel file, one sheet in the file, a row header cell, a column header cell, a data cell associated to them, and the location of the data cell.

4.5 Evaluation

From 20,743 Excel files collected, we selected at random 100 unseen files to evaluate the reliability of the extraction process; these files contained a total of 2,432 tables. To avoid very similar files in our evaluation, e.g., tables showing the same metric for distinct years, we

required the first three letters in the name of each newly selected file, to be different from the first three letters in the names of all the files previously selected to be included in the test batch. Given the naming convention at INSEE, this step reduces redundancy in the test set.

For these 100 files, we visually identified the header cells, data cells and header hierarchy, which we compared with those obtained from our extractor. We consider a table is "correctly extracted" when all these are pairwise equal; otherwise, the table is "incorrectly extracted". We recorded 91% (or 2,214) tables extracted correctly, and 9% (or 218) incorrectly extracted. Most of the incorrect extractions were due to sheets with several tables (not just one).

Overall, the extraction produced $2.68 \cdot 10^6$ row header cells, $122 \cdot 10^6$ column header cells, and $2.24 \cdot 10^9$ data cells; the extraction algorithm ran for 5 hours.

4.6 Implementation

The Excel files are collected with `insee-crawler`[11]. The crawler relies on Scrapy framework[12] to identify the downloadable links from `https://insee.fr`.

The main source code is written in Python 2 and it is open-sourced at `https://gitlab.inria.fr/cedar/excel-extractor`. All of the necessary libraries are specified in `requirements.txt`. The most important modules of this project are:

- *boundary_finder.py*: the algorithm to identify the boundary between multiple tables that appear in the same sheet.
- *data_model.py*: all the data structures used by other Python files.
- *data_provider.py*: extracts cells, rows, and columns from `Sheet` data structure.
- *table_extractor.py*: the extraction algorithm mentioned in Section 4.3.
- *test_extractor.sh*: runs the unit-test.

4.7 Related works

Closest to our work, [Chen and Cafarella, 2013] automatically extracts relational data from spreadsheets. Their system used conditional random field (CRF) [Lafferty et al., 2001] to identify *data frames*: such a frame is a *value region* which contains numeric cells, *top attributes* and *left attributes* corresponding to header cells and their hierarchies. From a collection of 410,554 Excel files collected from the Web, they selected randomly 100 files, labeled 27,531 non-empty rows (as title, header, data or footnote) manually by human experts, then used this dataset for training and evaluation. They obtained significant better precision and recall metrics when taking into account both textual and layout features for CRF. SVM was applied on header cells to learn about their hierarchies. The model achieved a precision of 0.921 for top headers, and 0.852 for left headers. However, these numbers should not be compared with our results, because the datasets are different (tables encountered on the Web *vs.* government statistics),

[11] `https://gitlab.inria.fr/cedar/insee-crawler`
[12] `https://scrapy.org/`

as well as the evaluation metrics (cell-by-cell assessment in [Chen and Cafarella, 2013] *vs.* binary assessment over the entire sheet extraction in our case). Overall, their method is more involved, whereas our method has the advantage of not requiring manual labeling. We may also experiment with learning-based approaches in the future. [Ahsan et al., 2016] describes an extractor to automatically identify in spreadsheets entities like location and time.

Data extraction from tables encoded as HTML lists has been addressed e.g., in [Elmeleegy et al., 2011]. Structured fact extraction from text Web pages has lead to the construction of knowledge bases such as Yago [Mahdisoltani et al., 2015] and Google's Knowledge Graph [Dong et al., 2014]. Our work has a related but different focus as we focus on high-value, high-confidence, fine-granularity, somehow-structured data such as government-issue statistics; this calls for different technical methods.

4.8 Conclusion and future works

We have described an effort to extract Linked Open Data from data tables produced by the INSEE statistics institute, and shared by them under the form of Excel tables. We do this as part of our work in the ContentCheck R&D project, which aims at investigating content management techniques (from databases, knowledge management, natural language processing, information extraction and data mining) to provide tools toward automating and supporting fact-checker journalists' efforts. The goal is to use the RDF data thus extracted as reference information, when trying to determine the degree of truthfulness of a claim.

We plan to experiment to spreadsheets from other sources to verify the generalization of our algorithm.

Chapter 5

Searching for truth in a database of statistics

5.1 Introduction

Statistic information available on the Web are not always easy to find. Some sites allow finding a dataset by keyword search; these sites' search engines are not always effective. Some sites invite users to navigate in a predefined hierarchy of categories in order to find the datasets; this works well when the users think in terms of these categories, but this is not always the case. Different organizations publish their datasets on Web sites structured differently; the structure of a given Web site may change with time etc. In Chapter 4, we have devised an approach to *extract* from high-quality, statistic Open Data in the "tables + text description" frequently used nowadays, *Linked* Open Data in RDF format.

In this chapter, we introduce novel algorithms for *searching for answers to keyword queries in a database of statistics*, organized in RDF graphs such as those we produced. First, we describe a *dataset search* algorithm, which given a set of user keywords, identifies the datasets (statistic table and surrounding presentations) most likely to be useful for answering the query. Second, we devised an *answer search* algorithm which, building on the above algorithm, attempts to answer queries, such as "unemployment rate in Paris in 2016", with *values* extracted from the statistics dataset, together with a *contextualization* quickly enabling the user to visually check the correctness of the extraction and the result relevance. In some cases, there is no single number known in the database, but several semantically close ones. In such cases, our algorithm returns the set of numbers, again together with context enabling its interpretation.

We have experimentally evaluated the efficiency and the effectiveness of our algorithms, and demonstrated their practical interest to facilitate fact-checking work. In particular, while political debates are broadcast live on radio or TV, fact-checkers can use such tools to quickly locate reference numbers which may help them publish live fact-checking material.

In the sequel, Section 5.2 defines the search problem we address, and describes our search algorithms. Section 5.3 presents our experimental evaluation. We briefly outline the software we developed in Section 5.4, and the related works in Section 5.5, then conclude (Section 5.6).

The content of this chapter was presented in a publication in the International Workshop on the Web and Databases 2018 [Cao et al., 2018b]. Together with the SBD 2017 work [Cao et al.,

2017], this has been informally published at BDA 2018 [Cao et al., 2018a].

5.2 Search problem and algorithm

	Seasonally adjusted youth (under 25s) unemployment				
	Number of persons (in thousands)				
	Oct-2016	Jul-2017	Aug-2017	Sep-2017	Oct-2017
Belgium	:	77	77	77	:
Bulgaria	27	22	21	19	19
Czech Republic	34	27	25	23	23
Denmark	62	54	53	49	47
Germany	293	283	283	283	283
France	663	629	625	623	625

Figure 5.1: Sample dataset on French youth unemployment.

Given a *keyword-based query*, we focus on returning a *ranked list of candidate answers*, ordered in the decreasing likelihood that they contain (or can be used to compute) the query result. A relevant candidate answer can be a data cell, a data row, column or even an entire dataset. For example, consider the query "youth unemployment in France in August 2017". An Eurostat dataset[1] is a good candidate answer to this query, since, as shown in Figure 5.1, it contains one data cell, at the intersection of the France row with the Aug-2017 column. Now, if one changes the query to ask for "youth unemployment in France in 2017", no single data cell can be returned; instead, all the cells on the France row qualify. Finally, a dataset containing 2017 French unemployment statistics over the general population (not just youth) meets some of the search criteria (2017, France, unemployment) and thus may deserve to appear in the ranked list of results, depending on the availability of better results.

This task requires the development of specific novel methods, borrowing ideas from traditional information retrieval, but following a new methodology. This is because our task is very specific: we are searching for information not within text, but within tables, which moreover are not flat, first normal form database relations (for which many keyword search algorithms have been proposed since [Hristidis and Papakonstantinou, 2002]), but partially nested tables, in particular due to the hierarchical nature of the headers, as we explained previously.

While most of the reasoning performed by our algorithm follows the two-dimensional layout of data in columns and tables, bringing the data in RDF have two advantages:

1. it puts a set of interesting, high-value data sources within reach of developers
2. it allows us to query across nested headers using regular path queries expressed in SPARQL (as we explain in Section 5.2.5)

We describe our algorithms for finding such answers below.

[1] http://ec.europa.eu/eurostat/statistics-explained/images/8/82/Extra_tables_Statis- tics_explained-30-11-2017.xlsx

5.2.1 Dataset search

The first problem we consider is: given a keyword query Q consisting of a set of keywords u_1,u_2,. . .,u_m and an integer k, find the k datasets most relevant for the query (we explain how we evaluate this relevance below).

We view each dataset as a *table* containing a *title*, possibly a *comment*, a set of *header cells* (row header cells and column header cells) and a set of *data cells*, the latter containing numeric data[2]. At query time, we transform the query Q into a set of of keywords $W = w_1, w_2, \ldots, w_n$ using the method described in Section 5.2.2. Offline, this method is also used to transform each dataset's text to words and we compute the score of each word with respect to a dataset, as described in Section 5.2.3. Then, based on the word-dataset score and W, we estimate datasets' relevance to the query as we explain in Section 5.2.4.

5.2.2 Text processing

Given a text t (appearing in a title, comment, or header cell of the dataset, or the text consisting of the set of words in the query Q), we convert it into a set of words using the following process:

- First, t is *tokenized* (separated into distinct words) using the KEA[3] tokenizers. Subsequently, each multi-word French location found in t that is listed in Geonames[4], is put together in a single token.
- Each token (word) is converted to lowercase, stop words are removed, as well as French accents which complicate matching.
- Each word is mapped into a word2vec vector, using the Gensim [Radim and Petr, 2010] tool. Bigrams and trigrams are considered following [Mikolov et al., 2013]. We had trained the word2vec model on a general-domain French news web page corpus (which consists of one million web pages) as our queries come from French news articles. When applying our system on a different context, e.g., tweets, we should find an appropriate corpus to train the word2vec model.

5.2.3 Word-dataset score

For each dataset extracted from the statistic database, we compute a score characterizing its semantic content. A first observation is that datasets should be returned not only when they contain the exact words present in the query, but also if they contain very closely related ones. For example, a dataset titled "Duration of marriage" could be a good match for the query "Average time between marriage and divorce" because of the similarity between "duration" and "time". To this effect, we rely on *word2vec* which provides similar words for any word in

[2]This assumption is backed by an overwhelming majority of cases given the nature of statistic data. We did encounter some counterexamples, e.g., http://ec.europa.eu/eurostat/cache/metadata/Annexes/mips_sa_esms_an1.xls. However, these are very few and thus we do not take them into account in our approach.

[3]https://github.com/boudinfl/kea

[4]http://www.geonames.org/

its vocabulary: if a word w appears in a dataset D, and w is similar to w', we consider w' also appears in D.

The score $score(w)$ of a dataset D w.r.t the query word w is 1 if w appears in D.

If w does not appear in D:

- If there exists a word w', from the list of top-50 similar words of w according to word2vec, which appears in D, then $score(w)$ is the similarity between w and w'. If there are several such w', we consider the one most similar to w.
- If w is the name of a Geonames place we can't apply the above scoring approach because "comparable" places (e.g., cities such as Paris and London) will have high similarity in the word2vec space. As the result, when user asks for "unemployment rate Paris", the data of London might be returned instead of Paris's. Let p be the number of places that Geonames' hierarchy API[5] returns for w (p is determined by Geonames and depends on w). For instance, when querying the API with *Paris*, we obtain the list *Île-de-France, France, Europe*. Let w'_i be the place at position i, $1 \leq i \leq p$ in this list of returned places, such that w'_i appears in D. Then, we assign to D a score for w equal to $(p+1-i)/(p+1)$, that is, the most similar place according to Geonames has rank $p/(p+1)$, and the least similar has the rank $1/(p+1)$. If D contains several of the places from the w's hierarchy, we assign to D a score for w corresponding to the highest-ranked such place.
- 0 otherwise.

Based on the notion of word similarity defined above, we will write $\boxed{w \prec W}$ to denote that the word w from dataset D either belongs to the query set W, or is close to a word in W. Observe that, by definition, for any $w \prec W$, we have $score(w) > 0$.

We also keep track of the *location(s)* (title, header and/or comment) in which a word appears in a dataset; this information will be used when answering queries, as described in Section 5.2.4.1. In summary, for each dataset D and word $w \in D$ such that $w \prec W$, we compute and store tuples of the form:

$$(w, score(w), location(w, D), D)$$

where $location(w, D) \in \{\texttt{T}, \texttt{HR}, \texttt{HC}, \texttt{C}\}$ indicates where w appears in D: `T` denotes the title, `HR` denotes a row header cell, `HC` denotes a column header cell, and `C` denotes an occurrence in a comment.

These tuples are encoded in JSON and stored in a set of files; each file contains the scores for *one* word (or bigram) w, and *all* the datasets.

5.2.4 Relevance score function

We now describe our score function, which relies on two sub-components: one content-based (Section 5.2.4.1) and one reflecting the location (Section 5.2.4.2).

[5] http://www.geonames.org/export/place-hierarchy.html

5.2.4.1 Content-based relevance score function

This function, denoted $g_1(D, W)$, quantifies the interest of dataset D for the word set W; it is computed based on the tuples $(w, score(w), location(w, D), D)$ where $w \prec W$.

We experimented with many score functions that give high ranking to datasets that have many matching keywords (see details in Section 5.3.2.2). These functions are monotonic in the score of D with respect to each individual word w. This enables us to apply Fagin's threshold algorithm (TA) [Fagin et al., 2003] to efficiently determine the k datasets having the highest g_1 score for the query W.

5.2.4.2 Location-aware score components

The location – title (T), row or column headers (HR or HC), or comments (C) – where a keyword occurs in a dataset can also be used to assess the dataset relevance. For example, a keyword appearing in the title often signals a dataset more relevant for the search than if the keyword appears in a comment. We exploit this observation to *pre-filter* the datasets for which we compute exact scores, as follows.

We run the TA algorithm using the score function g_1 to select $r \times k$ datasets, where r is an integer larger than 1. For each dataset thus retrieved, we compute a second, *refined* score function g_2 (see below), which also accounts for the locations in which keywords appear in the datasets; the answer to the query will consists of the top-k datasets according to g_2.

The second score function $g_2(D, W)$ is computed as follows. Let w' be a word appearing at a location $loc \in \{\mathtt{T}, \mathtt{HR}, \mathtt{HC}, \mathtt{C}\}$ such that $w' \prec W$. We denote by $w'_{loc,D}$ (or just w'_{loc} when D is clear from the context) the *existence of one or several located occurrence of* w' *in* D *in* loc. Thus, for instance, if "youth" appears twice in the title of D and once in a row header, this leads to two located occurrences, namely $\text{youth}_{\mathtt{T},D}$ and $\text{youth}_{\mathtt{HR},D}$.

Then, for $loc \in \{\mathtt{T}, \mathtt{HR}, \mathtt{HC}, \mathtt{C}\}$ we introduce a coefficient α_{loc} allowing us to calibrate the weight (importance) of keyword occurrences in location loc. To quantify D's relevance for W due to its loc occurrences, we define a *location score component* $f_{loc}(D, W)$. In particular, we have experimented with two f_{loc} functions:

- $f^{sum}_{loc}(D, W) = \alpha_{loc}^{\sum_{w \prec W} score(w_{loc,D})}$
- $f^{count}_{loc}(D, W) = \alpha_{loc}^{count\{w \prec W\}}$

where for $score(w_{loc,D})$ we use the value $score(w)$, the score of D with respect to w (Section 5.2.3). Thus, each f_{loc} "boosts" the relevance scores of all loc occurrences by a certain exponential formula, whose relative importance is controlled by α_{loc}.

Further, the relevance of a dataset increases if different query keywords appear *in different header locations*, that is, some in HR (header rows) and some in HC (header columns). In such cases, the data cells at the intersection of the respective rows and columns may provide very precise answers to the query, as illustrated in Figure 5.1: here, "France" is present in HC while "youth" and "17" appear in HR. To reflect this, we introduce another function $f_{\mathrm{H}}(D, W)$ computed on the scores of all unique located occurrences from row or column headers; we also experimented with the two variants, f^{sum}_{H} and f^{count}_{H} introduced above.

5.2.4.3 Content- and location-aware relevance score

Putting it all together, we compute the content- and location-aware relevance score of a dataset for W as:

$$g_2(D, W) = g_1(D, W) + \Sigma_{loc \in \{\texttt{T},\texttt{HR},\texttt{HC},\texttt{C}\}} f_{loc}(D, W) + f_{\texttt{H}}(D, W)$$

Finally, we also experimented with another function $g^*(D, W)$ defined as:

$$g_2^*(D, W) = \begin{cases} g_2(D, W), \text{if } f_{\texttt{T}}(D, W) > 0 \\ 0, \text{otherwise} \end{cases}$$

g_2^* discards datasets having no relevant keyword in the title. This is due to the observation that statistic dataset titles are typically carefully chosen to describe the dataset; if no query keyword can be connected to it, the dataset is very likely not relevant.

5.2.5 Data cell search

We now consider the problem of identifying the data cell(s) (or the data rows/columns) that can give the most precise answer to the user query.

Such an answer may consist of exactly one data *cell*. For example, for the query "unemployment rate in Paris", a very good answer would be a data cell $D_{r,c}$ whose closest row header cell contains "unemployment rate" and whose closest column header cell contains "Paris". Alternatively, query keywords may occur not in the closest column header cell of $D_{r,c}$ but in another header cell that is its ancestor in D. For instance, in Figure 5.1, let $D_{r,c}$ be the data cell at the intersection of the Aug-17 column with the France row: the word "youth" occurs in an ancestor of the Aug-17 header cell, and "youth" clearly characterizes $D_{r,c}$'s content. We say the closest (row and column) header cells of $D_{r,c}$ and all their ancestor header cells *characterize* $D_{r,c}$.

Another valid answer to the "unemployment rate in Paris" query would be a whole data *row* (or a whole *column*) whose header contains "unemployment" and "Paris". We consider this to be less relevant than a single data cell answer, as it is less precise.

We term *data cell answer* an answer consisting of either a data cell, or a row or column; below, we describe how we identify such answers.

We identify data cells answers from a given dataset D as follows. Recall that all located occurrences in D, and in particular those of the form $w_{\texttt{HR}}$ and $w_{\texttt{HC}}$ for $w \prec W$, have been precomputed; each such occurrence corresponds either to a header row r or to a header column c. For each data cell $D_{r,c}$, we define $\#(r, c)$ as the number of unique words $w \prec W$ occurring in the header cells characterizing $D_{r,c}$. Data cells in D may be characterized by:

1. Some header cells containing HR occurrences (for some $w \prec W$), and some others containing HC occurrences;
2. Only header cells with HR occurrences (or only header cells with HC ones).

Observe that if D holds both cell answers (case 1) and row- or column answers (case 2), by definition, they occur in different rows and columns. Our returned data cell answers from D are:

- If there are cells in case 1, then each of them is a data cell answer from D, and we return cell(s) with highest $\#(r, c)$ values.
- Only if there are no such cells but there are some relevant rows or columns (case 2), we return the one(s) with highest $\#(r, c)$ values. This is motivated by the intuition that if D has a specific, one-cell answer, it is unlikely that D also holds a less specific, yet relevant one.

Concretely, we compute the $\#(r, c)$ values based on the (word, score, location, dataset) tuples we extract (Section 5.2.3). We rely on SPARQL 1.1 [W3C,] queries on the RDF representation of our datasets to identify the cell or row/column answer(s) from each D. SPARQL 1.1 features property paths, akin to regular expressions; we use them to identify all the header cells characterizing a given $D_{r,c}$.

Note that this method yields only one element (cell, row or column) from each dataset D, or several elements if they have the exact same score. An alternative would have been to allow returning several elements from each dataset; then, one needs to decide how to collate (inter-rank) scores of different elements identified in different datasets. We consider that this alternative would increase the complexity of our approach, for unclear benefits: the user experience is often better when results from the same dataset are aggregated together, rather than considered as independent. Suggesting several data cells per dataset is then more a question of result visualization than one pertaining to the search method.

5.3 Evaluation

This section describes our experimental evaluation. Section 5.3.1 describes the dataset and query workload we used, which was split into a development set (on which we tuned our score functions) and a test set (on which we measured the performance of our algorithms). It also specifies how we built a "gold-standard" set of answers against which our algorithms were evaluated. Section 5.3.2 details the choice of parameters for the score functions. Finally, we present the prototype of our system in Section 5.3.3.

5.3.1 Datasets and queries

We collected all the articles published online by the fact-checking team "Les Décodeurs",[6] a fact-checking and data journalism team of Le Monde, France's leading national newspaper, between March 10th and August 2nd, 2014; there were 3,041 articles. From these, we selected 75 articles whose fact-checks were backed by INSEE data; these all contain links to `https://www.insee.fr`. By reading these articles and visiting their referenced INSEE dataset, we identified a set of 55 natural language queries which the journalists could have asked a system like ours[7]. Actually this data collection approach does not provide us all the articles that could be fact-checked by INSEE data. For example we could not spot the article about the unemployment rate but the journalist chooses OECD [8] data for fact-check it.

We experimented with a total of 288 variants of the g_2 function:

[6] `http://www.lemonde.fr/les-decodeurs/`

[7] This was not actually the case; our system was developed after these articles were written.

[8] `https://www.oecd.org/`

- g_1 was either g_{1a}, g_{1b} or g_{1c};
- g_2 relied either on f_{loc}^{sum} or on f_{loc}^{count}; for each of these, we tried different value combinations for the coefficients $\alpha_{\texttt{T}}$, $\alpha_{\texttt{HC}}$, $\alpha_{\texttt{HR}}$ and $\alpha_{\texttt{C}}$;
- we used either the g_2 formula, or its g_2^* variant.

We built a gold-standard reference to this query set as follows. We ran each query q through our dataset search algorithm for each of the 288 g_2 functions, asking for $k = 20$ most relevant datasets. We built the union of all the answers thus obtained for q and assessed the relevance of each dataset as either 0 (not relevant), 1 ("partially relevant" which means user could find some related information to answer their query) or 2 ("highly relevant" which means user could find the exact answer for their query); a Web front-end was built to facilitate this process.

5.3.2 Experiments

We specify our evaluation metric (Section 5.3.2.1), then describe how we tuned the parameters of our score function, and the results of our experiments focused on the quality of the returned results (Section 5.3.2.2). Last but not least, we put them into the perspective of a comparison with the baselines which existed prior to our work: INSEE's own search system, and plain Google search (Section 5.3.2.4).

5.3.2.1 Evaluation metric

We evaluated the quality of the answers of our runs and of the baseline systems by their mean average precision which is widely used for evaluating ranked lists of results.

MAP is traditionally defined based on a binary relevance judgment (relevant or irrelevant in our case). We experimented with the two possibilities:

- MAP_h is the mean average precision where only highly relevant datasets are considered as relevant
- MAP_p is the mean average precision where both partially and highly relevant datasets are considered relevant.

5.3.2.2 Parameter estimation and results

We experimented with the following flavors of the g_1 function :

- $g_{1b}(D, W) = 10^{\sum_{w \prec W} score(w)}$
- $g_{1d}(D, W) = 10^{count\{w \prec W\}}$
- $g_{1f}(D, W) = \sum_{w \prec W} 10^{score(w)}$

We also experimented with some modified variants that take into account the sum of matching keywords:

- $g_{1c}(D, W) = \sum_{w \prec W} score(w) + g_{1b}(D, W)$

	Dev. set 29 queries	Dev. set 17 queries
MAP_p	0.82	0.83
MAP_h	0.78	0.80

Table 5.1: Results on the first and second development set.

- $g_{1e}(D, W) = \sum_{w \prec W} score(w) + g_{1d}(D, W)$
- $g_{1g}(D, W) = \sum_{w \prec W} score(w) + g_{1f}(D, W)$

A randomly selected development set of 29 queries has been used to select the best values for the 7 parameters of our system : $\alpha_{\texttt{T}}$, $\alpha_{\texttt{C}}$, $\alpha_{\texttt{HR}}$, $\alpha_{\texttt{HC}}$, $\alpha_{\texttt{H}}$, as well as the different versions of g_1 and g_2. For this purpose, we ran a grid search with different values of these parameters, selected among $\{3, 5, 7, 8, 10\}$, on the development query set, and applied the combination obtaining the best MAP results on the test set (composed of the remaining 26 queries).

We found that a same score function has lead to the best MAP_h *and* the best MAP_p on the development query set. In terms of the notations introduced in Section 5.2.4, this best-performing score function is obtained by:

- Using $g_{1,c}$;
- Using f^{sum} and the coefficient values $\alpha_{\texttt{T}} = 10$, $\alpha_{\texttt{C}} = 3$, $\alpha_{\texttt{HR}} = 5$, $\alpha_{\texttt{HC}} = 5$ and $\alpha_{\texttt{H}} = 7$;
- Using the g_2^* variant, which discards datasets lacking matches in the title.

On the test set, this function has lead to $\boxed{MAP_p = 0.76}$ and $\boxed{MAP_h = 0.70}$.

Given that our test query set was relatively small, we performed two more experiments aiming at testing the robustness of the parameter selection on the development set:

- We used a randomly selected subset of 17 queries among the 29 development queries, and used it as a new development set. The best score function for this new development set was the same; further, the MAP results on the two development sets are very similar (see Table 5.1).
- We computed the MAP scores obtained on the full development set for all 288 combinations of parameters, and plotted them from the best to the worst (Figure 5.2; due to the way we plot the data, two MAP_h and MAP_p values shown on the same vertical line may not correspond to the same score function). The figure shows that the best-performing 15 combinations leads to scores higher than 0.80, indicating that any of these could be used with pretty good results.

We acknowledge the dataset is quite small despite the time consuming nature of the evaluation process. In the future work, we should spend more time to gather a bigger evaluation dataset in order to have a more robust evaluation.

5.3.2.3 Running time

Processing and indexing the words (close to those) appearing in the datasets took approximately three hours. We ran our experiments on a machine with 126GB RAM and 40 CPUs Intel(R)

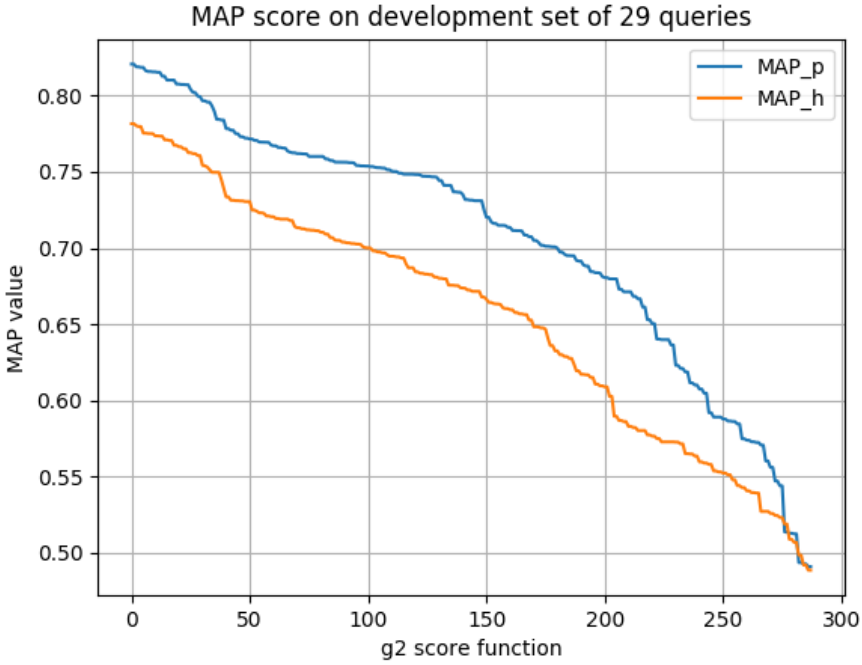

Figure 5.2: MAP results on the development set for 288 variants of the score function.

Xeon(R) E5-2640 v4 @ 2.40GHz. The average query evaluation time over the 55 queries we identified is 0.218 seconds.

5.3.2.4 Comparison against baselines

	Our system	INSEE search	Google search
MAP_p	0.76	0.57	0.76
MAP_h	0.70	0.46	0.69

Table 5.2: Comparing our system against baselines.

To put our results into perspective, we also computed the MAP scores of our test query set on the two baselines available prior to our work: INSEE's own dataset search system[9], and Google search instructed to look only within the INSEE web site. Similarly to the evaluation process of our system, for each query we selected the first 20 elements returned by these systems and manually evaluated each dataset's relevance to the given query. Table 5.2 depicts the MAP results thus obtained, compared against those of our system. Google's MAP is very close to ours; while our work is obviously not placed as a rival of Google in general, we view this as validating the quality of our results with (much!) smaller computational efforts. Further, our work follows a white-box approach, whereas it is well known that the top results returned by Google are increasingly due to other factors beyond the original PageRank [Brin and Page, 1998] information, and may vary in time and/or with result personalization, Google's own A/B testing etc.

We end this comparison with two remarks.

[9] Available at https://insee.fr

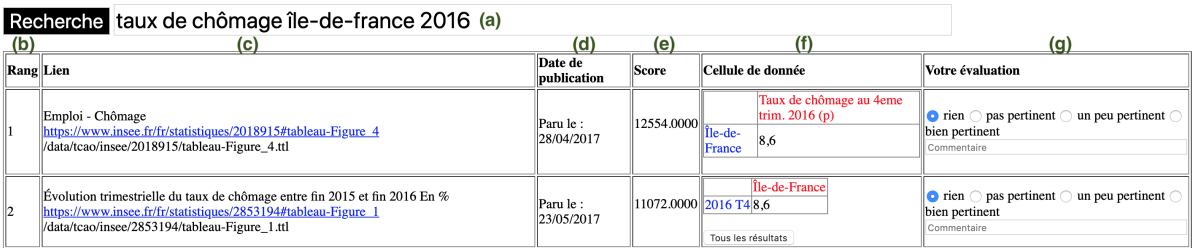

Figure 5.3: INSEE-Search web app

1. Our evaluation was made on INSEE data alone due to the institute's extensive database on which fact-checking articles were written, from which we derived our test queries. However, as stated before, our approach could be easily adapted to other statistic Web sites, as we only need the ability to crawl the tables from the Web site. As is the case for INSEE, this method may be more robust than using the category-driven navigation or the search feature built in the Web site publishing the statistic information.

2. Our system, based on a fine-granularity modeling of the data from statistic tables, is the only one capable of returning *cell-level answers* (Section 5.2.5). We show such answers to the users together with the header cells characterizing them, so that users can immediately appreciate their accuracy (as in Figure 5.3).

5.3.3 Web application for online statistic search

We develop a web app (available at `http://statsearch.inria.fr`) to let users experience and evaluate our system. A simple account registration is asked before using the web app. The app is illustrated in Figure 5.3:

- **(a)** is the text box to let user input their search query
- **(b)** is the rank of search results
- **(c)** contains the datasets' title, the link of the original dataset and the location of the extracted RDF data.
- **(d)** is the published date of the dataset
- **(e)** is the relevance score of each dataset
- **(f)** is the relevant data cells found in each dataset, the red and blue texts represent the matched headers
- **(g)** is the section to let user evaluate the search result as *not relevant*, *partially relevant*, and *highly relevant*.

5.4 Implementation

The source code is written in Python 3 and it is open-sourced at `https://gitlab.inria.fr/cedar/excel-search`. We have developed our system in Python (61 classes and 4071

lines). All of the necessary libraries are specified in `requirements.txt`. The following list describes the most important modules:

- *src/algorithm/fagin.py*: implements the Fagin algorithm.
- *src/algorithm/document_ranking.py*: implements the algorithm in Section 5.2.
- *src/webapp/app.py*: the Flask[10] app which renders system's web interface (Section 5.3.3).
- *src/word_document_score.py*: computes the word-dataset score (Section 5.2.3).
- *src/test/*: contains the unit-test files.

5.5 Related works

In this chapter, we focused on improving the usability of statistic tables (HTML tables or spreadsheets) as reference data sources against which claims can be fact-checked. Other works focused on building textual reference data source from general claims [Levy et al., 2014, Bar-Haim et al., 2017], congressional debates [Thomas et al., 2006] or tweets [Rajadesingan and Liu, 2014].

Knowledge graphs comprising *facts*, e.g., "the capital of Germany is Berlin", and *semantic rules*, e.g., "the president of a country is also a citizen of that country", can also be used as reference sources for fact-checking, as discussed in Section 3.2. Such knowledge graphs mostly contain "general knowledge" and of textual and relational nature (entities and relations which hold between them), whereas the statistics we focus on are mostly numerical. Thus, we view our work as a useful complement to the construction of reference knowledge bases. Early predecessors of Web-based knowledge base construction focused on extracting and integrating structured data from Web HTML tables and lists [Elmeleegy et al., 2009, Cafarella et al., 2009], and our data extraction work (Chapter 4) performs a similar task. However, extraction from statistic spreadsheets brings specific challenges in particular due to the frequent hierarchical headers. [Han et al., 2008] describes a tool for producing RDF graphs from Google spreadsheets, with the help of a mapping specification where users describe the RDF structure in which various spreadsheet cells should be exported. Our extraction is less flexible but at the same time is easier to use as it requires zero input from users; if needed, one can restructure the RDF we produce by means of RDF-specific tools. Finally, some work focused on building a knowledge base or graph embeddings (with a representation similar to word2vec) [Ristoski and Paulheim, 2016, Grover and Leskovec, 2016], but to the best of our knowledge, none of them have represented jointly unstructured textual content and structured bases.

Some works focused on exploiting the data in HTML and spreadsheet tables found on the Web. Tschirschnitz et al. [Tschirschnitz et al., 2017] focused on detecting the semantic relations that hold between millions of Web tables, for instance detecting so-called *inclusion dependencies* (when the values of a column in one table are included in the values of a column in another table). Closest to our work, M. Kohlhase et al. [Kohlhase et al., 2013] built a search engine for finding and accessing spreadsheets by their formulae. This is less of an issue for the tables we focus on, as they contain plain numbers and not formulas.

[10] http://flask.pocoo.org/

Google's Fusion Tables work [Gonzalez et al., 2010] focuses on storing, querying and visualizing tabular data, however, it does not tackle keyword search with a tabular semantics as we do, nor is it concerned with keyword search. Google has also issued Google Tables as a working product[11]. In March 2018, we tried to use it for some sample queries we addressed in this paper, but the results we obtained were of lower quality (some were irrelevant). We believe this may be due to Google's focus on data available on the Web, whereas we focus on very high-quality data curated by INSEE experts, but which needed our work to be easily searchable.

5.6 Conclusion and future works

We have presented an efficient search algorithm to retrieve, from the RDF corpus we build out of INSEE data, the dataset (or, when possible, even the exact cell values) that are most pertinent for a given keyword search. This improves the usability of this high-quality data. Our system can be used to quickly get reference information with respect to a given claim.

Currently, our software is not capable of aggregating information, e.g., if one asks for unemployed people from all departments within a region, we are not capable of summing these numbers up into the number corresponding to the region. This could be addressed in future work which could focus on applying OLAP-style operations of drill-down or roll-up to further process the information we extract from the INSEE datasets.

[11] https://research.google.com/tables

Chapter 6

Extracting statistical mentions from textual claims to provide trusted content

6.1 Introduction

In this chapter, we describe the last missing step of our system: the **extraction of claims referring to statistical mentions from text sources**. This step allows to automatically formulate the search queries which our system (Chapter 5) can solve against the RDF corpus we gathered in Chapter 4. Our whole system can help fact-checking journalists as it automates the task of finding checkable claims in massive text sources, and of bringing up the closest reference data source value for the given claim. Based on these, the journalists can choose the truth label which seems most appropriate, write an interpretation of the result and share it with their readers.

Beyond fact-checking textual Web sources, the application which motivated the work of this thesis, the technique we describe also serves to link text data sources, at the fine granularity of sentences, to statistical entities.

Firstly, we outline our approach in Section 6.2 and describe its details in Section 6.3. We evaluate our algorithm in Section 6.4. The implementation of our system is presented in Section 6.5. We present the related works in Section 6.6 following by conclusions and perspectives (Section 6.7).

The content of this chapter was presented in a publication in the International Conference on Applications of Natural Language to Information Systems 2019 [Cao et al., 2019c].

6.2 Statistical claim extraction outline

Figure 6.1 outlines the method we devised for *identifying and extracting statistical claims from text*. The figure also shows how the three main contributions of this thesis hold together: at the bottom, the red modules designate the result of the extraction described in Chapter 4, respectively, the search algorithm outlined in Chapter 5.

From the publication context of statistic data (the text in header of statistics tables) we extract

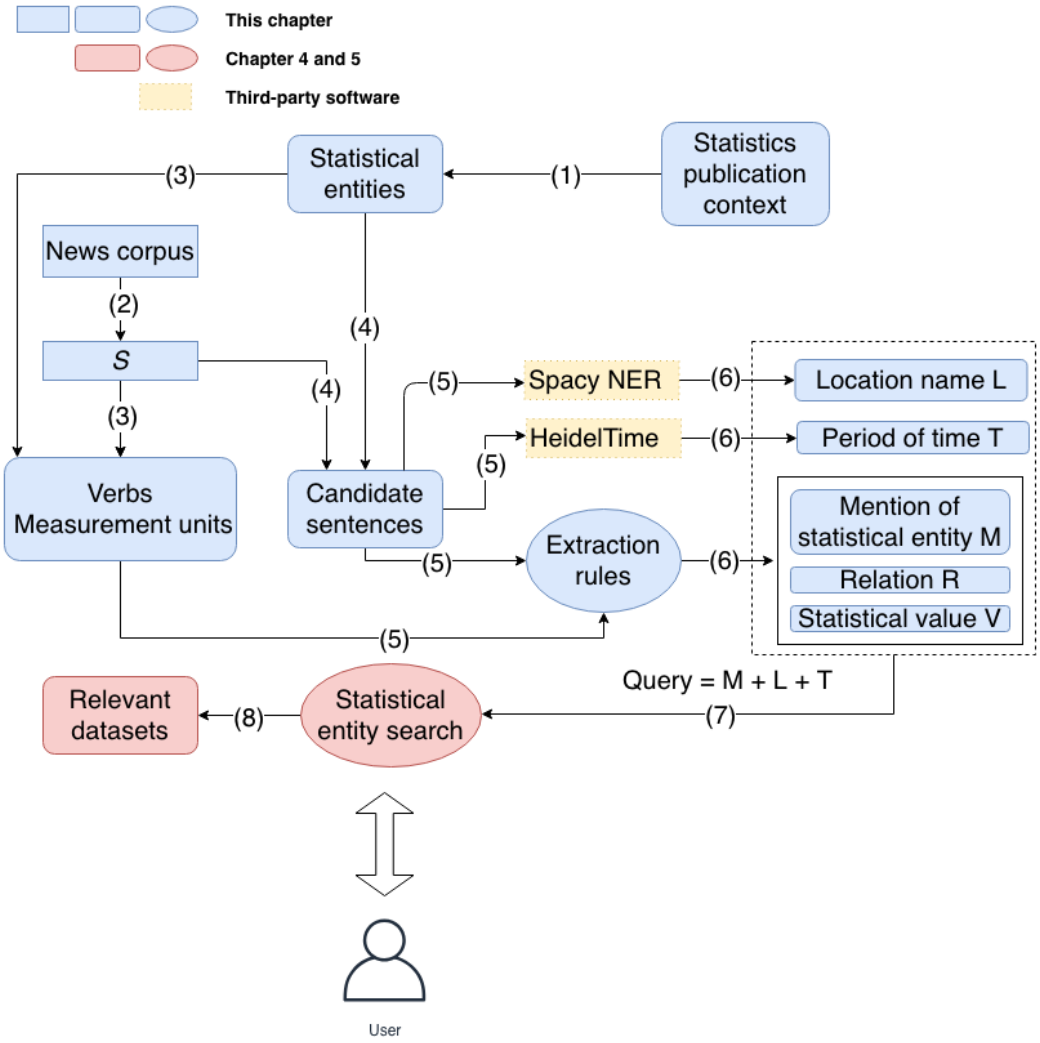

Figure 6.1: Main processing steps of our statistical claim extraction method.

a set of **statistical entities** (step (1) in the figure), those whose reference values are known in the statistic dataset for some time periods and/or geographical area, such as *"unemployment", "youth unemployment", "unemployment in Aquitaine in 2015", "gross domestic product"*. From 111,145 tables published by INSEE, we have obtained a total of 1,397 statistic entities, as we detail in Section 6.3.1.

Then, we have built a **text corpus** which we selected with an interest in topics that INSEE studies. We focused on news articles, and because most INSEE metrics refer to the economy domain, we looked for articles on such topics. We collected news articles between 2010 and 2018 from the three following newspapers, using the NewsPlease software [Hamborg et al., 2017]:

1. Articles from Le Monde and Le Figaro have explicit URIs which contain the broad topic to which the article belongs. Thus, we required "economie", "economie-francaise" and/or "emploi" appear in the URIs of Le Monde articles, and similarly "flash-eco", "economie", "emploi" for Le Figaro.

2. Articles from Les Echos do not have such an URI component and are by and large all on economic topics. Thus, we selected the candidate articles by running the LDA (Latent Dirichlet allocation) [Blei et al., 2003] topic modelling algorithm to generate 20 topics. We then reviewed these topics manually and picked the most relevant ones[1].

From these articles, we have extracted (step (2)) 322,873 sentences containing at least one numerical value. From now on, we will refer to these sentences as S. From S, we extract (step (3)) all the verbs which state a numerical value, e.g., "amounts to", "is worth", "decreases" etc., as well as all the measurement units, e.g., "people", "euros", "percentage", "point" (sometimes used as an alternative to "percentage") etc.

Next, we identify among S sentences the **candidate sentences** which could claim a relationship between a statistical entity and a value. This is done (step (4)) by selecting those S sentences which mention statistic entities. From each candidate sentence, e.g., *"France's public debt fell slightly, by 11.4 billion euros, between the second and third quarters of 2013"*, we extract:

- a mention of statistical entity M, e.g., *public debt*;
- a location L, e.g., *France*, by extracting geographical places using the spaCy Named Entity Recognition (NER) tool[2]; L may be missing from the sentence.
- a time period T, e.g., *2013*, which is extracted using HeidelTime [Strötgen and Gertz, 2010]; T may also be missing;
- a relation R, e.g., *fell*, connecting M to V in the sentence. R may also be missing, e.g., in a phrase such as *"France's 60 million inhabitants..."*;
- a statistical value V, e.g., *"11.4 billion euros"*.

The approach we devised to extract the components M, R and V is described in Section 6.3. We present an evaluation of the performance of the extraction algorithm in Section 6.4.1.

[1] All topics and their keywords are available at https://gitlab.inria.fr/tcao/news-scraper/blob/master/lesechos_topics_all.txt. The topics we selected for this work are those numbered 1, 2, 3, 7 as they contain many economic keywords

[2] https://spacy.io/models/fr#fr_core_news_md

For each (M, L, T, R, V) tuple extracted as above, the (M, L, T) query is generated (step (7)) and sent to our keyword search algorithm (Section 5.2). We omit R in the query since the purpose of extracting R is to confirm the relationship between M and V. The search results are evaluated in section 6.4.2.

6.3 Entity, relation and value extraction

In this section, we describe the details of our extraction approach. Section 6.3.1 outlines the pre-processing steps we perform to obtain a list of statistical entities. Section 6.3.2 shows how we collect a list of verbs and a list of measurement units that help identifying candidate sentences. In Section 6.3.3, we describe an attempt to identify the relationship between M and V using bootstrapping approach. Finally, Section 6.3.4 outlines the extraction rules which lead to obtaining the relevant M, R and V components.

6.3.1 Statistical entities

We made a hypothesis of the existence of statistical entities in the headers of statistic tables. For example, one header of table[3] is *"Taux de chômage au T1 2015"* ("Unemployment rate in the first quarter of 2015"). We apply the following text processing steps on the statistics publication context (in our case, the statistic table headers):

- Pick only headers that contain measurement unit such as euro, %, etc. These headers are usually noun phrases in format *Entity + (Unit)* such as "Unemployment rate in 2015 (in %)". We prefer to rely on table headers and not on table titles and comments, since the latter are longer sentences that could (or could not) contain the entities, and customarily do contain and much more irrelevant information. Further, we require that headers contain a measurement unit because one is present in any statistical entity measure, thus we can use that as an indicator of potentially interesting phrases.
- Filter out the measurement unit: in the above example, this leads to the snippet "Unemployment rate in 2015".
- Filter out possible date time values and the associated prepositions; this leaves us with the statistical entity "Unemployment rate".
- A final manual filtering allowed us to weed out some text snippets which do not in fact comprise relevant entities.

Through the above process, we obtained 1,397 statistical entities, some of which are presented, together with their frequencies from the statistics publication context, in Table 6.1. Note that they are all related to quite hot topics in current public policy debates, in particular in Europe and in the US, making them useful and interesting for fact-checking.

[3] https://www.insee.fr/fr/statistiques/1288156#tableau-Figure_2

Extracted statistical entities	Frequency
intensité de la pauvreté (intensity of poverty)	190
nombre d'entreprises (number of companies)	176
taux de pauvreté au seuil de 60% (poverty rate at 60% median wages)	130
chômeurs (unemployed people)	104
taux de chômage (unemployment rate)	79
excédent brut d'exploitation (Earnings before Interest, Taxes and Amortization)	68
PIB (gross domestic product, GDP)	54
taux de population en sous-emploi (share of people working less than they would like)	54
solde migratoire (net migration)	44
taux de marge (margin rate)	28
taux de pauvreté (poverty rate)	21
taux de mortalité (mortality rate)	15
population en sous-emploi à temps partiel (population working part-time, willing to work more)	12

Table 6.1: Sample extracted statistical entities.

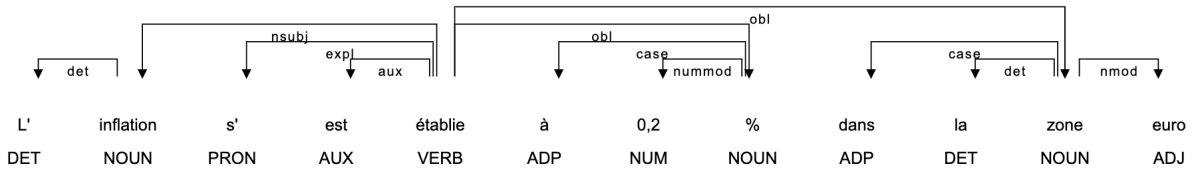

Figure 6.2: Sample dependency tree built by spaCy.

6.3.2 Relevant verbs and measurement units

We use the annotation S_I to refer to the candidate sentences that contain the word *"insee"*. These sentences are likely to feature a relationship between a mention of statistical entity M as a noun phrase (e.g., "unemployment rate") and a statistical value V as a numerical value, optionally followed by a measurement unit (e.g., "5%"). We use the spaCy parser [Honnibal and Johnson, 2015] to build the dependency tree of sentence; Figure 6.2 illustrates one example. On the line at the bottom, spaCy states the morphological (part-of-speech) category it recognized for each word (token). Thus, NOUN designates nouns, VERB a verb, AUX an auxiliary verb, NUM a number, ADJ an adjective etc. Sometimes, numerical values in French contain whitespaces, e.g. *19 000*, which the dependency tree represents as several nodes. We remove these white spaces in order to have one single NUM (numerical token in spaCy parlance) node, e.g., "19000". The arrows between tokens trace their syntactic dependencies, e.g., the "euro" adjective at the end of the phrase characterizes (depends on) the "zone" word preceding it.

A *syntactical dependency path* in S_I of the form $(w_0{:}P_0, d_0, w_1{:}P_1, d_1, w_2{:}P_2, \ldots)$ where each w_m is a word in the sentence, P_m is the part-of-speech assigned to the word by spaCy and d_m is the label of an edge between P_m and P_{m+1}; a path can follow an edge either in its direction, or in the opposite direction. For instance, (zone:NOUN, `amod`, euro:ADJ) is a path of length 1

in Figure 6.2, so is its reverse (euro:ADJ, amod, zone:NOUN), whereas (établie:VERB, obl, %:NOUN, nummod, 0,2:NUM) is a path of length 2.

We collected the syntactic dependency paths connecting M, R and V as follows. For each NOUN node, we located the paths that connect it to a NUM node. We found many paths that start with (NOUN, nsubj, VERB) (a noun is subject of a verb); we refer to them as $Paths_I$. In Figure 6.2, the path connecting *"inflation"* to *"0,2"* is (inflation:NOUN, nsubj, établie:VERB, obl, %:NOUN, nummod, 0,2:NUM).

As the relation R of M and V is generally introduced by specific verbs, we collected all the verbs associated with VERB nodes from $Paths_I$. To make sure of the quality of the collected verbs, we filtered manually from the original list to retain 129 relevant ones; in the sequel, we denote them by *I_verbs*.

Based on $Paths_I$, we also gathered a set of measurement units by collecting all the NOUN nodes connected to a NUM node via a nummod edge (numeric modifier). We call this list *I_units*.

6.3.3 Bootstrapping approach

If a dependency path p between M and V appears many times in different sentences, p could be a good indicator to signal the relationship between the mention of statistical entities and the statistical value. With that hypothesis, we collect the sentences that contain statistical entity extracted from section 6.3.1. From these sentences, we identify the corresponding dependency path p and then retain the paths that have the highest frequencies. Let's call these paths $Paths_B$.

Given the input sentence i, we search for the existence of any dependency path from $Paths_B$. If such path exists, we are able to locate the nodes that represent M and V respectively. This approach, inspired by [Saha et al., 2017], has been evaluated but turned out to be less robust than the extraction rules described below, with similar performance on development set but much lower on test set.

6.3.4 Extraction rules

Given the input sentence i and a statistical entity e, we extract the mention of statistical entity M, the statistical value V and their relation R. If there is no relationship between e and the statistical value, or there is no statistical value in i, we return the value $M = None$.

Firstly we generate the dependency tree $t(i)$ of i using spaCy, as illustrated in Figure 6.2. Then we identify in this tree the statistical entity e and the numerical value(s), as follows.

1. We filter out the year values (e.g., 2018) since we only want to search for the relationship of statistical entity and statistical value.
2. We define the distance $d(n_1, n_2)$ of two nodes n_1 and n_2 in $t(i)$ as the absolute value of n_1's position - n_2's position. For instance, $d($*inflation*, *établie*$) = 3$.
3. The distance $D(e, v)$ from e to a numerical value v is the minimum value of $d(e$'s first word, $v)$ and $d(e$'s last word, $v)$. In case there are more than one numerical values, we select the one that has the smallest $D(e, v)$ as the statistical value of e.

4. We identify the dependency path $p(i)$ that connects **the first word of** e (let's call it s) and e**'s statistical value** (if available), let's call it n. With our sample dependency tree, $p(i) =$ `nsubj`, établie:VERB, `obl`, %:NOUN, `nummod`, 0,2:NUM)

5. We look for the node u directly connected to n (the last one before n) in $p(i)$. If u is a noun and there is a `nummod` edge between u and n, we return $M =$ *None* in the following cases:

   - u does not appear in *I_units*.
   - u appears in *I_units* and in the input sentence, there is an article or a preposition between s and u.

   On the contrary, we extract the relevant nodes from:

   (a) the first NOUN node s: we identify the nodes that connect to s via `nmod` and `amod` (adjectival modifier) edges, then we collect subtree of these nodes.

   For example, when the input i is *"Il se poursuit avec le nombre de demandeurs d'emplois de moins de 25 ans, qui a reculé de 2,6% en France métropolitaine sur un mois en septembre."* ("It continues with the number of job seekers under the age of 25, who fell by 2.6% in metropolitan France in September"), e is *"nombre de demandeurs d'emplois"* (number of job seekers), s is *"nombre"* (number). We identify the node *"demandeurs"* that connects to s via `nmod` edge. The nodes extracted in this case are *"nombre, de, demandeurs, de, emplois, de, moins, de, 25, ans"* (number of job seekers under the age of 25).

   As a second example, consider the input sentence *"Cette baisse qui intervient après 5 mois consécutifs de hausse permet de ramener l'inflation française en dessous de la barre des 2 % en rythme annuel."* ("This decline, which comes after five consecutive months of increase, makes it possible to reduce French inflation below the 2% mark on an annual basis."). In this case both e and s are *"inflation"*; we identify the node *"française"* that connects to s via `amod` edge. The nodes extracted in this case are *"inflation, française"* (French inflation).

   (b) the VERB node *verb*: the subtree of nodes that connect to *verb* via `obl` edge (a nominal dependent of a verb). We impose the constraints of having *verb* appear in *I_verbs* and the leftmost node of the subtree must be a preposition among *"en, à, dans"*. For example, let the input sentence be *"En décembre, l'inflation s'est établie à 0,2 % dans la zone euro, faisant courir à l'économie le risque de sombrer dans l'atonie."* ("In December, inflation stood at 0.2% in the euro zone, putting the economy at risk of falling into sluggishness."), e and s are *"inflation"*. We identify the node *"zone"* that connects to the VERB node *"établir"* through a `obl` edge. The nodes extracted in this case are *"inflation, dans, la, zone, euro"* (inflation in the euro zone).

   If the nodes from these subtrees appear in $p(i)$, we do not include them.

All the extracted nodes form the mention of statistical entity M. The statistical value V is composed of n and u. The relation R is composed the nodes from $p(i)$ which do not belong to M and V.

6.4 Evaluation

We evaluated our system at two levels. First, Section 6.4.1 evaluates the quality the extraction rules and relevance of the search results. Second, Section 6.4.2 evaluates the performance of the end-to-end system.

6.4.1 Evaluation of the extraction rules

We select some statistical entities [*"taux de chômage" (unemployment rate), "nombre de demandeurs d'emploi" (number of job seekers), "niveau de vie" (life's quality), "consommation des ménages" (household consumption), "PIB" (gross domestic product, GDP), "inflation" (inflation), "SMIC" (minimum wages), "taux d'emploi" (employment rate)*] from the list of statistical entities (Section 6.3.1). For each entity e we pick randomly 50 sentences that contain e then we split randomly 25 sentences for development set and 25 sentences for test set. Finally there are 200 sentences for each set. If there is no relationship between e and the statistical value, or there is no statistical value in the given sentence, we assign a label *NoStats*. Otherwise we annotate each sentence with e and the relevant phrases (we call these phrases *contexts* of e) to form a mention of statistical entity.

For a given sentence, if the extraction rules return $M = None$ and we have the *NoStats* label from the annotated sentence then the extraction is an accurate one. On the contrary, we verify if the extracted M contains e and one of its contexts. In that case, the extraction is also accurate. We compute the accuracy of our extraction rules in both development set and test set and obtain the scores of 71.35% and 69.63% respectively.

6.4.2 Evaluation of the end-to-end system

We selected randomly 38 sentences for the test set (from which 26 were considered as extracted correctly at previous step – Section 6.4.1). We gave the corresponding generated queries $q = M + L + T$ as input to the INSEE-Search system (Section 5.2).

We evaluated the accuracy of the system using the same metric from Section 5.3.2.1. Note that there is no guarantee that any "highly relevant" element at all exists in the dataset for each query.

The results are available in Table 6.2 and show that, given an arbitrary claim (related to statistic entities), fine-grained and relevant information can be returned in the vast majority of the cases. They also show that, as in all keyword-based search systems, building a perfect query is neither necessary or sufficient for obtaining good results. Even if a good entity extraction improves the results, we can still find highly or partially relevant information even if the entity extraction is not perfectly achieved.

These findings are now to be confirmed by an evaluation on more claims, more databases and based on a real-user study. We also showed in Section 5.2 that the performance of our query system was similar to a document-level search engine such as Google, but with a much better granularity of the information (data cell instead of web page).

		$MAP_h(10)$	$MAP_p(10)$
Overall performance (38 sentences)		**0.672**	**0.789**
among which	M extracted correctly (26)	0.725	0.829
	M extracted incorrectly (12)	0.559	0.703

Table 6.2: Evaluation of INSEE-Search.

6.5 Implementation

The source code is written in Python 3 and it is open-sourced at `https://gitlab.inria.fr/cedar/statstical_mentions`. The following list describes the most important modules:

- *data* contains the annotated data
- *candidate_sentences.csv* contains the candidate sentences
- *Evaluation NLDB.ipynb* contains the extraction rules and their evaluation

6.6 Related works

BONIE [Saha et al., 2017] claimed to be the first open numerical relation extractor. Given the input sentence "Hongkong's labour force is 3.5 million", BONIE extracts the triple (Hong Kong; has labour force of; 3.5 million). They create high precision patterns to extract seed facts from input sentences. Seed fact has the following format *(entity head-word, relation head-word, quantity, unit)*. And then they apply bootstrapping to increase the number of seed facts: searching for sentence from a text corpus that contain all words in a seed fact. The next step is pattern learning. For each candidate sentence, they use placeholder {entity}, {relation}, {quantity} to represent the match of entity, relation, quantity + unit. Then they search for the minimum path in the dependency tree that connect these 3 placeholders. All the paths are sorted by frequency and the top-1000 paths were used to evaluate system's performance. Finally they have some rules to select which subtree in the dependency tree that they want to collect in order to obtain the entity and relation. When trying their approach for our settings, we found that the learnt pattern were sometimes too generic or too specific and they failed to capture the correct dependency path in the new texts.

ClausIE [Corro and Gemulla, 2013] is an open information extraction system. It first detects clauses in a sentence and then apply specific rules for each type of clause in order to extract the entity of interest. ClausIE also relies on a hand-crafted dictionary of verbs to identify the existence of relation in sentence. Compare to their approach, we have a "semi-automated" solution to identify the list of verbs.

ClaimBuster [Hassan et al., 2017] was the first work on check-worthiness. They used annotated sentences from US election debates to train a SVM classifier in order to determine whether or not a sentence is a check-worthy claim. This is the common approach when having a large amount of training data, which is not the case in French.

Platypus [Pellissier Tanon et al., 2018] receives natural language query as input and deliver its answers from Wikidata. The system generates logical representations of the given input using

a set of transformation rules or template based technique (which requires training data). The transformation rules operate on the set of part-of-speech tags and dependency tags. E.g., the dependency "Where $\rightarrow$ nsubj $\rightarrow$ X" could be transform to $\{p|(X, locatedIn, p)\}$. Regarding the template based technique, free variable of logical representation is annotated in natural language questions. They use Conditional Random Fields algorithm to recognize entities in the query and fill these entities into the logical representation. Finally Platypus ranks these representations to retrieve the most relevant ones and execute SPARQL query on Wikidata to retrieve the answer. Compare to them, we do not depend on predefined rules to identify the connection between entities of interest.

AggerChecker [Jo et al., 2019] translates natural language claims to SQL queries. Firstly, a set of keywords from relational tables is collected. For each standard SQL aggregation function, such as COUNT, SUM, etc., they associate it with a set of keywords, e.g., "total", "number" for COUNT. They also collect keywords from table names and its columns. All of the collected keywords are stored in an information retrieval engine [4] in order to retrieve the relevant matches with respect to claims' keywords. For each input text, they identify a set of keywords with some simple heuristics rules. They also take into account users' inputs in order to select relevant keywords. Finally, they use an expectation maximization model to compute a distribution over SQL queries for each claim. The inputs of this model are the queries constructed from the matches of the information retrieval engine and the associated score of each match. This paper is a very similar work to our end-to-end system which returns relevant datasets to answer to the extracted claims from textual content. In their system, the quality of keyword extraction from tables could pose a problem because people tend to use short words or even abbreviations to name columns and table names. In our system, spreadsheets aim to serve the public audience so they usually have meaningful names on their tables.

BriQ [Ibrahim et al., 2019] identifies the corresponding table cells of textual quantities mentions appearing in the same document. They train a binary classifier to determine the alignment of a pair of mention and cell. The results of this classifier provide a list of candidate pairs. The aggregation appearing in mention, for example sum, average, etc., is detected by another classifier in order to decide whether we should map a mention to a single cell or the aggregation of several cells. As a human would tries to identify the correct pair by spotting neighboring quantities in both text and tables, they build a graph to model the relationships of mentions and cells. A random walk with restart algorithm [Lao et al., 2011] is applied on this graph to compute the probability of having an alignment between a mention and a cell. This probability and the confidence scores from the two previous classifiers form an overall score to determine the alignment. This work focuses on identifying a pair of statistic value and the corresponding value which appears in a table. In our work, we focus on finding a mention of statistic entity and its relationship with a statistic value. BriQ and AggerChecker [Jo et al., 2019] bring us some ideas for the future works of detecting the aggregation of statistic values.

6.7 Conclusion and future works

In this chapter we have presented an end-to-end system for identifying statistic claims and finding in a statistic database the relevant statistic data for checking this claim. The main steps for this system are:

[4] Apache Lucence https://lucene.apache.org/

1. converting the statistic database into an RDF graph (Chapter 4),
2. identifying all statistic entities relevant for this database,
3. selecting the check-worthy claims mentioning these entities,
4. converting the claim into a set of keyword,
5. using an keyword search, focused algorithm to extract the relevant pieces of data (ideally, table cells) relevant to the claim (Chapter 5).

To apply our system on a different statistic database, we only need reuse our extraction system (Chapter 4) for step 1 and then pick the statistical entities as mentioned in Section 6.3.1 for step 2. The remaining steps remain unchanged. To make the RDF graph up-to-date, our crawler works on a daily basis to collect the latest statistic tables. We also leave journalists state whether the claim is ”true”, ”mostly true”, ”mostly false” etc. This is because they insist that translating a relative error (difference between the reference and the stated value) into a truth judgment depends on the context and the measure, and they absolutely want to make that call themselves.

A classic defect of these pipeline approaches in NLP systems is that errors accumulate at each step. Nevertheless, our results show that we often manage to find useful information for the user, which will make the human work of fact-checking easier and faster. For the future work, we want to be able to identify the *implicit* statistical claims, for example “The unemployment rate this year is higher than last year’s.”.

Chapter 7

Topics exploration and classification from newspapers and social media

This chapter describes an effort we made to build an interesting corpus for journalistic fact-checking. Instead of consisting of statistic data, as was the case of the INSEE corpus on which the work of Chapter 4, 5 and 6, this corpus is about *facts, statements and beliefs*, and it is organized as an RDF knowledge base. The classes and properties of the corpus have been defined in another PhD work, also part of ContentCheck, that of Ludivine Duroyon (U. Rennes 1). The effort carried as part of my thesis focused on extracting, from text and semistructured data, information that can be used to fill in such a graph. This effort contributed to a joint demonstration co-authored with Ludivine and her advisors [Cao et al., 2019a, Cao et al., 2019b].

Background: a model for facts, statements and beliefs We recall here the model introduced in [Duroyon et al., 2019], to model statements that different individuals and organizations make, and capture their spread through time. The model defines a set of RDF classes and properties, and specifies how they can be combined to describe a data graph. In this model, **facts** are assumed to hold irrespectively of the actors' viewpoints. In contrast, a **belief** is held by an actor (individual or organization), and it has a **subject**, which can be *positive* (the agents believes something; this is denoted by a plus sign $+$) or negative (the agent does not believe it, denoted by a minus sign $-$). Facts, beliefs, and communications can be endowed with a begin and an end time.

All these components of the model are illustrated in Figure 7.1, where for readability, we omitted the property names (edges on the label). The data in Figure 7.1 is organized around five central nodes: the facts F_1 and F_2, a belief B_1, and two communications C_1 and C_2. Their attached time information (begin and end) is encoded in the timestamp resources t_0 to t_4, each of which has a beginning and an end. The fact F_1 has a description d_1, which states that "Gilets Jaunes" protest against fuel prices in Bordeaux on Dec 1, 2018. Any kind of nodes/resources may be used to describe a fact, depending on its nature. The statement "E. Macron has believed from November 26th, 2018, to November 30th, 2018, that Yellow vests[1] will not protest against fuel prices in Bordeaux, France; on December 1st, 2018." is encoded by the (negative) belief B_1 of the agent (human user icon) "E. Macron", whose subject is F_1. The fact F_2 is an increase of minimum wages, announced by E. Macron; note that F_2 is not is

[1]Name of a social movement started in France in November 2018, see https://en.wikipedia.org/wiki/Yellow_vests_movement.

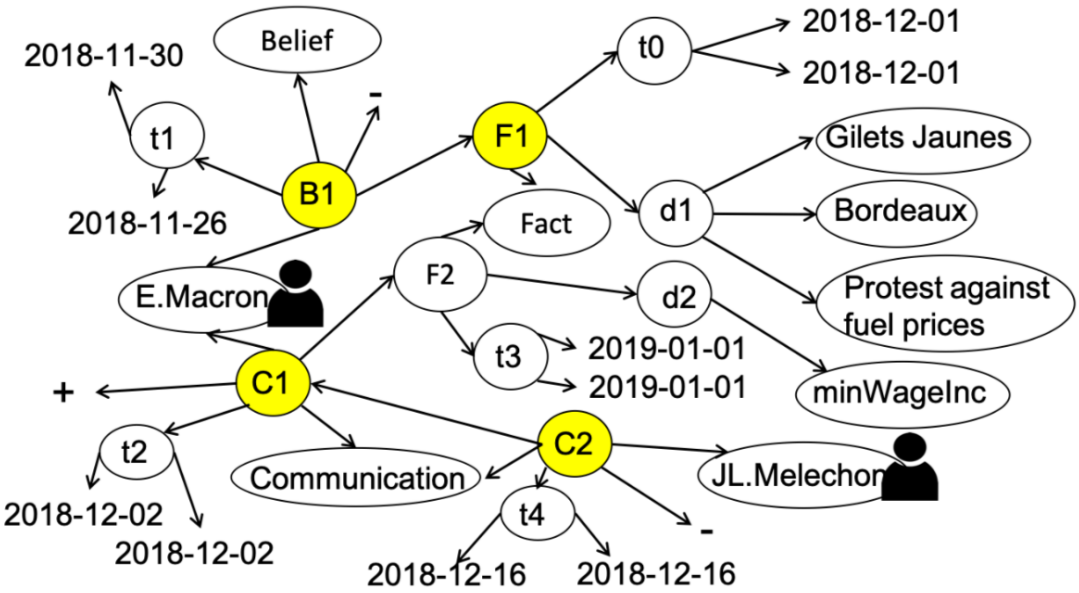

Figure 7.1: Sample facts and beliefs

not recorded as certainly true (trusted by the database), but only as "according to" E. Macron. Formally, [Duroyon et al., 2019] distinguishes a set of resources which are of type Record, that is, believed (considered true) by the database; in Figure 7.1, these are the yellow nodes. Finally, C_1 and C_2 are communications; specifically, C_2 is a communication of Jean-Luc Mélenchon stating that the communication C_1 did not happen (note the $-$ sign of C_2).

The work described in this chapter focused on *instantiating* this model starting from real-world data. Specifically:

1. We built a corpus of tweets and news articles as described in Section 7.1.
2. Then, we developed a topic extraction module (Section 7.2) to identify the topics present in the tweets.
3. Finally, we developed a weakly supervised neural classifier (Section 7.3) to attach communication (tweets) to their subjects (considered to be the topics identified by the previous topic extraction module), then conclude (Section 7.4).

7.1 Corpus construction

We collected 96,513 French tweets containing the words "gilets jaunes" (yellow vests) using the Twint tool [2]. We denote by C_t the corpus built from these tweets. From these, we extracted the HTTP links they contained (omitting those that refer to social networks such as Facebook and Youtube, as well as Wikipedia). We fed the links thus obtained to the `NewsPlease` [Hamborg et al., 2017] news article crawler/extractor, leading to a set of 56,657 news articles.

[2] https://github.com/twintproject/twint

On each article, we apply the previously developed *source extractor*[3] module to extract the source (name of individual or organization who makes a statement) in each sentence, when this is possible. For example, in the sentence "M. Macron blames the violence of Yellow Vests demonstrators", `source-extractor` could identify "M. Macron" as a source. This leads to a total of 76,566 sentences, which form a corpus we denote C_s.

7.2 Topic extraction

We rely on Scholar algorithm [Card et al., 2018] to extract a set of topics where each topic is represented by a set of relevant keywords. The Scholar algorithm uses an *autoencoding* neural network. An auto-encoder is a feed-forward neural network, whose last layer has as many nodes as the input, and whose goal is to *reconstruct its input*. Specifically, an autoencoder has two main parts:

- The *encoder* takes an input vector x from a space $\mathcal{X}$ and maps it into another space $\mathcal{Z}$; this is called the *latent representation* of x;
- The *decoder* takes the $\mathcal{Z}$ representation of an input x and maps it back into an $x' \in \mathcal{X}$.

The training objective of an autoencoding network is to minimize a *reconstruction error* $L(x, x')$. The latent representations are oftentimes more compact than the input, and can thus be seen as *compressed versions* thereof.

Following this auto-encoder approach, [Card et al., 2018] use a neural network layer as the latent representation; for L they rely on the Kullback–Leibler divergence [Kullback and Leibler, 1951], which is approximated with the Evidence Lower Bound formula [Kingma and Welling, 2014].

We introduce the basic notions and notations used in [Card et al., 2018], then explain the encoder (inference model) and the decoder (generative model). Finally, we described the extracted topics.

Notions [Card et al., 2018] introduces the following notions:

- D is the collection of all documents, where each document is viewed as a collection of words.
- V is the vocabulary of all words, denoted $v_1, v_2, \ldots, v_N$.
- K is the number of topics defined by users.
- $freq(w)$ is the number of occurrences of the word w in D.
- w is a document from D, viewed as a list of words.

Inference model [Card et al., 2018] defines $\pi = f_e(x)$ where f_e is a multi-layer perceptron and x is a word-count vector over V in w.

Then, they compute μ and σ as linear transformations of π respectively.

The latent representation r is computed as $r = \mu + \sigma \odot \epsilon$ where $\odot$ denotes the element-wise product operator and $epsilon = \mathcal{N}(0, 1)$ (a normal distribution).

[3] https://gitlab.eurecom.fr/asrael/source-extractor

Generative model They feed r to a softmax layer (a layer that applies softmax function[4] on output of the previous layer) to obtain the document representation θ. Then they pass θ to a linear layer to obtain the topic-word distribution $\eta = \theta^T B + d$, where

- $B^{K \times N}$ is a weight matrix;
- $d^{K \times N}$ is also a matrix, $d_i = [\log(freq(w_1)), \log(freq(w_2)), \ldots, \log(freq(w_N))]$ is a vector which contains the log of the overall word frequency for $i \in \{1, \ldots, K\}$

To extract the top-m relevant words with respect to a topic i, they collect the m words with highest values from the ith row of η.

Topic extraction results An implementation of the algorithm proposed in [Card et al., 2018] is released at `https://github.com/dallascard/scholar`. Running this algorithm on the two corpora C_s, respectively C_t, with $K = 20$, results in 14 topics for C_s and 18 topics for C_t[5]. The keywords of these topics are available online[6].

7.3 Topic classification

Training a supervised deep learning model is dependent on the availability of a large corpus of labeled data. To work around the difficulty of building such a corpus manually, we rely on the WESTClass (Weakly Supervised Neural Text Classification) method [Meng et al., 2018]. We introduce the notions and parameters of WESTClass and explain how to use the von Mises Fisher distribution to represent a class (Section 7.3.1). Then, we describe the details of training a WESTClass model (Section 7.3.2). Finally, we integrate it to our settings and report the evaluation results (Section 7.3.3).

7.3.1 Preliminaries

[Meng et al., 2018] introduce the following notions and parameters:

- D is the corpus containing all the documents, each of which is a collection of words.
- m is the number of classes into which we want to classify the documents. The model will output a probability distribution over m classes for each document.
- $k_{j1}, k_{j2}, \ldots$ are sets of *seed keywords*, supplied by users, to describe the semantics of a class. For this purpose, we use the topics identified as explained in Section 7.2.
- V is the vocabulary that contains all the words available in D.
- $freq(w)$ is the frequency of the word w in D.
- v_w is the embedding of the word w.
- β is the number of documents that we want to generate for the class j.

[4] $\text{softmax}(x_{t'}) = e^{x_{t'}} / \sum_t e^{x_t}$

[5] We reviewed the keywords associated with each topic to retain the relevant topics

[6] `https://gitlab.inria.fr/tcao/westclass/blob/master/keywords/tweet_keywords.txt`

- α, a real number in the range $(0, 1)$, is used to control the construction of pseudo documents and labels (see below).
- γ, an integer, is a parameter to control the construction of pseudo documents.
- t, an integer, is the number of keywords that represent a class.

They create m von Mises Fisher (vMF) [Banerjee et al., 2005, Gopal and Yang, 2014] distributions for the m classes. They collect more seed keywords for each class j by computing the average similarity between the embeddings of words from V and j's seed keywords, then choosing the most similar ones. After this process, each class j has t seed keywords out of which to construct a vMF distribution vMF_j that represents the relevant words with respect to j.

Pseudo document and pseudo label They define the background distribution $p_b(w)$ of the word w over V as:

$$p_b(w) = \frac{freq(w)}{\sum_{w \in D} freq(w)}$$

Each document vector $d_i, i \in 1, 2, \ldots, \beta$ is sampled from the distribution vMF_j. Let V_{d_i} be the set of γ words whose embeddings are the most similar with d_i's embeddings. They generate each word w for the pseudo document D_i^* with the following probability distribution:

$$p(w|d_i) = \begin{cases} \alpha p_b(w) & \text{if } w \notin V_{d_i} \\ \alpha p_b(w) + (1-\alpha)\frac{exp(d_i^T v_w)}{\sum_{w' \in V_{d_i}} exp(d_i^T v_{w'})} & \text{otherwise} \end{cases}$$

They define the pseudo label l_{ij}, which is the probability that D_i^* belongs to the class j. It is computed using the formula:

$$l_{ij} = \begin{cases} (1-\alpha) + \alpha m & \text{if } D_i^* \text{ is generated from class } j \\ \alpha/m & \text{otherwise} \end{cases}$$

7.3.2 Model training

Using pseudo documents and pseudo labels, they train a neural network that minimizes the Kullback–Leibler divergence distance [Kullback and Leibler, 1951] (also called the relative entropy) between the outputs of the network and the pseudo labels. Experiments in [Meng et al., 2018] rely on two neural network architectures: CNN and Hierarchical Attention Network (HAN) [Yang et al., 2016]. This step is called **pre-training**.

Since the pre-trained model is trained on the pseudo documents, it could not generalize well on the real documents from D. To tackle that issue, they perform another step called **self-training**. The model predicts labels for documents from D. After each n iterations, it recomputes l_{ij} as $\frac{y_{ij}^2/f_j}{\sum_k y_{ik}^2/f_k}$ where f_k is the number of pseudo documents having their predicted label as k and y_{ij} is the current predicted probability that D_i^* belongs to the class j. The model's weights are updated to minimize the Kullback-Leiber divergence distance as mentioned preciously. The model also compares current and predicted labels to compute the percentage of documents that have their labels changed. If this percentage is smaller than a pre-defined threshold σ, the self-training step could be stopped. If the σ threshold is not reached, the self-training process is stopped after N iterations, where N is a pre-defined constant.

	F1, Dev	F1, Test	MAP@3, Dev	MAP@3, Test
C_s	0.727	0.711	0.738	0.721
C_t	0.785	0.817	0.823	0.842

Table 7.1: Quality of the classification.

7.3.3 Evaluation

Using the topics (keyword sets) identified in Section 7.2 as inputs of WESTClass, we chose the CNN architecture for pre-training and self-training. The performance is evaluated on two datasets: 192 randomly chosen sentences from C_s and 162 randomly chosen tweets from C_t. For each dataset, we use 50% of the data for development set and the remaining data for test set. We perform grid search to find the best hyperparameters[7] for CNN model and report the best results in the Table 7.1.

7.4 Conclusion

We have described our approach to identify the relevant topics present in tweets data without the need of large amount of training data. The identified topics are used to instantiate an RDF model that captures the statements made by individuals and organizations and their spread through time.

[7] available at https://gitlab.inria.fr/tcao/westclass

Chapter 8

Conclusion

We present a summary of the thesis in Section 8.1 and discuss the perspectives in Section 8.2.

8.1 Summary

We developed an end-to-end system to identify factual information from textual content and to provide relevant data. Our datasets and source code are available publicly. We also let people try our system at https://statsearch.inria.fr/. This system contains three modules:

1. an *extractor* that crawls statistic spreadsheets and transforms them to RDF datasets
2. a *search engine* that identifies relevant RDF datasets with respect to a user query
3. a *claim detector* that detects statistical claims from texts

The extractor leverages the visual and semantic cues of spreadsheets to identify locations of header and data cells. Then extracted cells are populated in an RDF conceptual data model. As the spreadsheets are issued from the official data sources, we believe that they are credible to verify against misinformation.

The search engine quantifies the relevance of each dataset by its header cells' content. The location (title, headers, or comments) where a keyword occurs in a dataset is also taken into account to compute the dataset relevance. We search the relevant datasets efficiently using Fagin algorithm. To let users quickly analyzing a matched result, we identify the data cells that contain the relevant data with respect to user's query. The functionalities of this search engine help to save a lot of time for end users since they have quick access to pertinent information.

The last module of our system, the claim detector, offers a solution to find statistical claims using linguistic cues and extraction rules. The extracted claim is converted to an input of the search engine. This module allows us to integrate with the two above-mentioned modules as an end-to-end system for verifying information.

On a joint work, we develop a system with the capability of identifying popular topics on social network. This system enable us to understand people's viewpoints about current social movements.

During the thesis, we had tried some other research directions but their results were not promising. For the search engine, we computed a sentence embedding model of all sentences that contain numerical values from INSEE. To retrieve the relevant sentences with respect to a given sentence, we compute its embedding vector and obtain the vectors that are close to it in the embeddings vector space. This approach produces a list of semantic relevant sentences but it fails to retrieve statistical data to answer to the input query. We also tried some machine learning approaches for the claim detector module. Firstly, we annotated statistical entities and trained an Named Entity Recognition model. As the size of our training set was small, the model failed to obtain a good accuracy. So we made use of crowdsourcing to enlarge our dataset. This time we encountered the problem of low quality annotations. Our description for the annotators was not well designed which led to the misunderstand of some annotators. After reading the Snorkel paper [Ratner et al., 2017], we implemented some labeling functions to generate training set. We did not obtain a reliable accuracy measurement on the validation set even after seeking advice from the Snorkel team. Then we came back to the idea of building the training set manually but this time we applied some active learning models in order to reduce the training set's size by selecting only the informative examples. The obtained results of these experiments were not good as we expected.

8.2 Perspectives

Driven by the French journalism context which framed the work described in this thesis, we focused on data sources of obvious fact-checking interest (notably INSEE statistics, as well as tweets which were often mentioned as a source of interesting information by journalists from Les Décodeurs). This has also driven the language processing and information extraction part of this work to be devised for French. We view this as a strong point of our work, given the relative scarcity of tools and linguistic techniques for non-English languages, in particular French.

We consider to generalize our algorithms to other statistical data sources, for example OECD [1], French public data [2], etc. We will need to verify the accuracy of our extraction system (Chapter 4) on these new sources and make our extraction algorithm more flexible. We could also investigate the extraction of other data types (e.g., text or speech). For example we could perform text analysis on government statements or reports in order to collect facts such as government spending, inflation rate, etc. Finally, we should also try to implement our system with other languages, for example English. This task involves specific customizations for each specific language.

Currently, we identify claims in text using rule-based approach. In order to make our system more efficient, we should consider to examine text's *check-worthiness*, i.e., the task of classifying whether a sentence is worthy to fact-check. We are collaborating with journalists at Le Monde[3] to develop a training dataset for this task. We could also try to build an Entity Recognition model using an annotated corpus to see whether it could identify mentions of statistical entities that do not appear in our pre-defined list. Our system currently can not detect these mentions.

[1] https://data.oecd.org/
[2] https://www.data.gouv.fr/
[3] https://www.lemonde.fr/

We apply many linguistic tools on our system, for example syntactical dependency tree, named-entity recognition, date time recognition. The integration of all these tools make the analysis complex and prone to errors. If we improve the accuracy of each tool then the overall performance of the whole system would be higher. We could also experiment with new tools such as *coreference resolution*, i.e., finding the word/phrase that refer to the same entity across sentences, in order to identify all instances of statistical mentions. With the success of transfer learning techniques for NLP recently, we believe that these techniques could help us to improve the accuracy of many downstream NLP tasks.

We should also look for other data sources that could be utilized for fact-checking task. An example is the fact-checked articles from journalists which have been published under the ClaimReview[4] standard. We plan to build a system that matches these articles against a user query such as *"Has the government removed teacher jobs to fund supplementary police officers?"*[5].

Another interesting research direction is analyzing the temporal dimension of statistical data, for example finding data to assess whether the unemployment rate has been rising in the last three years. This task involves identifying the published time attribute from our RDF datasets and organizing it efficiently in order to query the evolution of data through time.

Finally, the fact-checking results should be able to engage and entertain their audience. Some possible approaches are live fact-check during political debate, interactive data visualization that let users explore the relevant data. On another hand, the interaction from end-users could provide us evaluation data and feedback to improve our system's performance as well as developing new useful features.

[4] https://schema.org/ClaimReview

[5] https://www.lemonde.fr/les-decodeurs/article/2019/06/03/le-gouvernement-a-t-il-supprime-des-postes-de-profs-pour-financer-des-policiers-supple
5470810_4355770.html

Bibliography

[Ahmadi et al., 2019] Ahmadi, N., Lee, J., Papotti, P., and Saeed, M. (2019). Explainable fact checking with probabilistic answer set programming. *CoRR*, abs/1906.09198.

[Ahsan et al., 2016] Ahsan, R., Neamtu, R., and Rundensteiner, E. (2016). Towards spreadsheet integration using entity identification driven by a spatial-temporal model. In *ACM SAC*, pages 1083–1085, New York, NY, USA. ACM.

[Bach et al., 2017] Bach, S. H., Broecheler, M., Huang, B., and Getoor, L. (2017). Hinge-loss markov random fields and probabilistic soft logic. *J. Mach. Learn. Res.*, 18(1):3846–3912.

[Bahdanau et al., 2015] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Baird et al., 2017] Baird, S., Sibley, D., and Pan, Y. (2017). Talos Targets Disinformation with Fake News Challenge Victory.

[Banerjee et al., 2005] Banerjee, A., Dhillon, I. S., Ghosh, J., and Sra, S. (2005). Clustering on the unit hypersphere using von mises-fisher distributions. *J. Mach. Learn. Res.*, 6:1345–1382.

[Bar-Haim et al., 2017] Bar-Haim, R., Bhattacharya, I., Dinuzzo, F., Saha, A., and Slonim, N. (2017). Stance Classification of Context-Dependent Claims. In *EACL*, pages pages 251–261.

[Barrón-Cedeño et al., 2018] Barrón-Cedeño, A., Elsayed, T., Suwaileh, R., Màrquez, L., Atanasova, P., Zaghouani, W., Kyuchukov, S., Da San Martino, G., and Nakov, P. (2018). Overview of the CLEF-2018 CheckThat! Lab on Automatic Identification and Verification of Political Claims. Task 1: Check-Worthiness. *CEUR Workshop Proceedings*, 2125.

[Berners-Lee et al., 1998] Berners-Lee, T., Fielding, R., and Masinter, L. (1998). Uniform resource identifiers (uri): Generic syntax.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

[Brants et al., 2004] Brants, S., Dipper, S., Eisenberg, P., Hansen-Schirra, S., König, E., Lezius, W., Rohrer, C., Smith, G., and Uszkoreit, H. (2004). Tiger: Linguistic interpretation of a german corpus. *Research on Language and Computation*, 2(4):597–620.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Mach. Learn.*, 45(1):5–32.

[Breunig et al., 2000] Breunig, M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). Lof: Identifying density-based local outliers. In *PROCEEDINGS OF THE 2000 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA*, pages 93–104. ACM.

[Brin and Page, 1998] Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117.

[Cafarella et al., 2009] Cafarella, M. J., Halevy, A. Y., and Khoussainova, N. (2009). Data integration for the relational web. *PVLDB*, 2(1):1090–1101.

[Cao et al., 2019a] Cao, T. D., Duroyon, L., Goasdoué, F., Manolescu, I., and Tannier, X. (2019a). BeLink: Querying Networks of Facts, Statements and Beliefs (demonstration). In *Bases de Données Avancées*.

[Cao et al., 2019b] Cao, T. D., Duroyon, L., Goasdoué, F., Manolescu, I., and Tannier, X. (2019b). BeLink: Querying Networks of Facts, Statements and Beliefs (demonstration). In *International Conference on Information and Knowledge Management*.

[Cao et al., 2017] Cao, T. D., Manolescu, I., and Tannier, X. (2017). Extracting linked data from statistic spreadsheets. In *Proceedings of The International Workshop on Semantic Big Data*, SBD '17, pages 5:1–5:5, New York, NY, USA. ACM.

[Cao et al., 2018a] Cao, T. D., Manolescu, I., and Tannier, X. (2018a). Extracting Linked Data from statistic spreadsheets. In *Bases de Données Avancées*.

[Cao et al., 2018b] Cao, T.-D., Manolescu, I., and Tannier, X. (2018b). Searching for truth in a database of statistics. In *Proceedings of the 21st International Workshop on the Web and Databases*, WebDB'18, pages 4:1–4:6, New York, NY, USA. ACM.

[Cao et al., 2019c] Cao, T.-D., Manolescu, I., and Tannier, X. (2019c). Extracting statistical mentions from textual claims to provide trusted content. In *24th International Conference on Applications of Natural Language to Information Systems*.

[Card et al., 2018] Card, D., Tan, C., and Smith, N. A. (2018). Neural Models for Documents with Metadata. *ACL*.

[Carlson et al., 2010] Carlson, A., Betteridge, J., Wang, R. C., Hruschka, Jr., E. R., and Mitchell, T. M. (2010). Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 101–110, New York, NY, USA. ACM.

[Castillo et al., 2011] Castillo, C., Mendoza, M., and Poblete, B. (2011). Information credibility on twitter. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 675–684, New York, NY, USA. ACM.

[Cazalens et al., 2018] Cazalens, S., Lamarre, P., Leblay, J., Manolescu, I., and Tannier, X. (2018). A content management perspective on fact-checking. In *WWW (Companion Volume)*, pages 565–574. ACM.

[Chen and Cafarella, 2013] Chen, Z. and Cafarella, M. (2013). Automatic web spreadsheet data extraction. In *Proceedings of the 3rd International Workshop on Semantic Search Over the Web*, Semantic Search, pages 1:1–1:8, New York, NY, USA. ACM.

[Conneau et al., 2017] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language

inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

[Corro and Gemulla, 2013] Corro, L. D. and Gemulla, R. (2013). ClausIE : Clause-Based Open Information Extraction. In *WWW 2013 - Proceedings of the 22nd International Conference on World Wide Web*, number i, pages 355–365.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

[Cramer, 2002] Cramer, J. (2002). The Origins of Logistic Regression. Tinbergen Institute Discussion Papers 02-119/4, Tinbergen Institute.

[Cunningham and Delany, 2007] Cunningham, P. and Delany, S. J. (2007). k-nearest neighbour classifiers.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Dong et al., 2014] Dong, X. L., Gabrilovich, E., Heitz, G., Horn, W., Murphy, K., Sun, S., and Zhang, W. (2014). From data fusion to knowledge fusion. *PVLDB*, 7.

[Duroyon et al., 2019] Duroyon, L., Goasdoué, F., and Manolescu, I. (2019). A Linked Data Model for Facts, Statements and Beliefs. In *International Workshop on Misinformation, Computational Fact-Checking and Credible Web*, WWW '19 Companion - Proceedings of the 2019 World Wide Web Conference, San Francisco, United States.

[Elmeleegy et al., 2011] Elmeleegy, H., Madhavan, J., and Halevy, A. (2011). Harvesting relational tables from lists on the web. *The VLDB Journal*, 20(2):209–226.

[Elmeleegy et al., 2009] Elmeleegy, H., Madhavan, J., and Halevy, A. Y. (2009). Harvesting relational tables from lists on the web. *PVLDB*, 2(1):1078–1089.

[Fagin et al., 2003] Fagin, R., Lotem, A., and Naor, M. (2003). Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):pages 614–656.

[Ferrara et al., 2017] Ferrara, A., Montanelli, S., and Petasis, G. (2017). Unsupervised detection of argumentative units though topic modeling techniques. In Habernal, I., Gurevych, I., Ashley, K. D., Cardie, C., Green, N., Litman, D. J., Petasis, G., Reed, C., Slonim, N., and Walker, V. R., editors, *ArgMining@EMNLP*, pages 97–107. Association for Computational Linguistics.

[Firth, 1957] Firth, J. R. (1957). A synopsis of linguistic theory 1930-55. 1952-59:1–32.

[Frakes and Baeza-Yates, 1992] Frakes, W. B. and Baeza-Yates, R. (1992). *Information Retrieval Data Structures & Algorithms*. Prentice Hall, Englewood Cliffs, New Jersey.

[Friedman, 2000] Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

[Friedman et al., 1997] Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Mach. Learn.*, 29(2-3):131–163.

[Gencheva et al., 2017] Gencheva, P., Nakov, P., Màrquez, L., Barrón-Cedeño, A., and Koychev, I. (2017). A context-aware approach for detecting worth-checking claims in political debates. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 267–276, Varna, Bulgaria. INCOMA Ltd.

[Goasdoué et al., 2013] Goasdoué, F., Karanasos, K., Katsis, Y., Leblay, J., Manolescu, I., and Zampetakis, S. (2013). Fact checking and analyzing the web (demonstration). In *ACM SIGMOD*.

[Gonzalez et al., 2010] Gonzalez, H., Halevy, A., Jensen, C., Langen, A., Madhavan, J., Shapley, R., and Shen, W. (2010). Google fusion tables: Data management, integration, and collaboration in the cloud. In *SOCC*.

[Gopal and Yang, 2014] Gopal, S. and Yang, Y. (2014). Von mises-fisher clustering models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages I–154–I–162. JMLR.org.

[Graves, 2018] Graves, L. (2018). Understanding the Promise and Limits of Automated Fact-Checking. *Factsheet*, (February):1–7.

[Gray et al., 2012] Gray, J., Chambers, L., and Bounegru, L. (2012). *The Data Journalism Handbook: How Journalists can Use Data to Improve the News*. O'Reilly.

[Gray et al., 2007] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., and Pirahesh, H. (2007). Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *CoRR*, abs/cs/0701155.

[Grover and Leskovec, 2016] Grover, A. and Leskovec, J. (2016). Node2Vec: Scalable Feature Learning for Networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864, New York, NY, USA. ACM.

[Hamborg et al., 2017] Hamborg, F., Meuschke, N., Breitinger, C., and Gipp, B. (2017). news-please: A generic news crawler and extractor. In Gaede, M., Trkulja, V., and Petra, V., editors, *Proceedings of the 15th International Symposium of Information Science*, pages 218–223.

[Han et al., 2008] Han, L., Finin, T., Parr, C., Sachs, J., and Joshi, A. (2008). Rdf123: From spreadsheets to rdf. In Sheth, A., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., and Thirunarayan, K., editors, *International Semantic Web Conference (ISWC)*, pages 451–466, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Hansen et al., 2019] Hansen, C., Hansen, C., Alstrup, S., Simonsen, J. G., and Lioma, C. (2019). Neural Check-Worthiness Ranking with Weak Supervision: Finding Sentences for Fact-Checking. *MisinfoWorkshop2019*, 2.

[Hassan et al., 2017] Hassan, N., Zhang, G., Arslan, F., Caraballo, J., Jimenez, D., Gawsane, S., Hasan, S., Joseph, M., Kulkarni, A., Nayak, A. K., Sable, V., Li, C., and Tremayne, M. (2017). Claimbuster: The first-ever end-to-end fact-checking system. *Proc. VLDB Endow.*, 10(12):1945–1948.

[Heilman and Smith, 2009] Heilman, M. and Smith, N. A. (2009). Question generation via overgenerating transformations and ranking. Technical Report CMU-LTI-09-013, Carnegie Mellon University.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[Honnibal and Johnson, 2015] Honnibal, M. and Johnson, M. (2015). An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366.

[Howard and Ruder, 2018] Howard, J. and Ruder, S. (2018). Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

[Hristidis and Papakonstantinou, 2002] Hristidis, V. and Papakonstantinou, Y. (2002). DISCOVER: keyword search in relational databases. In *Very Large Databases Conference (VLDB)*, pages 670–681.

[Ibrahim et al., 2019] Ibrahim, Y., Riedewald, M., Weikum, G., and Zeinalipour-Yazti, D. (2019). Bridging quantities in tables and text. *Proceedings - International Conference on Data Engineering*, 2019-April:1010–1021.

[Jo et al., 2019] Jo, S., Trummer, I., Yu, W., Wang, X., Yu, C., Liu, D., and Mehta, N. (2019). Verifying Text Summaries of Relational Data Sets. pages 299–316.

[Joachims, 1999] Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 200–209, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Jones, 1972] Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21.

[Joulin et al., 2016] Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., and Mikolov, T. (2016). Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

[Kang et al., 2014] Kang, J. S., Feng, S., Akoglu, L., and Choi, Y. (2014). ConnotationWordNet: Learning connotation over the Word+Sense network. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1544–1554, Baltimore, Maryland. Association for Computational Linguistics.

[Karadzhov et al., 2017] Karadzhov, G., Nakov, P., Marquez, L., Barron-Cedeno, A., and Koychev, I. (2017). Fully Automated Fact Checking Using External Sources. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*.

[Karimi and Tang, 2019] Karimi, H. and Tang, J. (2019). Learning Hierarchical Discourse-level Structure for Fake News Detection. *NAACL*.

[Kilicoglu et al., 2012] Kilicoglu, H., Shin, D., Fiszman, M., Rosemblat, G., and Rindflesch, T. C. (2012). Semmeddb: a pubmed-scale repository of biomedical semantic predications. *Bioinformatics*, 28(23):3158–3160.

[Kim, 2014] Kim, Y. (2014). Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

[Kingma and Welling, 2014] Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. *ICLR*, (Ml):1–14.

[Klema and Laub, 1980] Klema, V. and Laub, A. (1980). The singular value decomposition: Its computation and some applications. *IEEE Transactions on Automatic Control*, 25:164–176.

[Kohlhase et al., 2013] Kohlhase, M., Prodescu, C., and Liguda, C. (2013). Xlsearch: A search engine for spreadsheets. *EuSpRIG*.

[Konstantinovskiy et al., 2018] Konstantinovskiy, L., Price, O., Babakar, M., and Zubiaga, A. (2018). Towards Automated Factchecking: Developing an Annotation Schema and Benchmark for Consistent Automated Claim Detection. *EMNLP*, pages 1–18.

[Kullback and Leibler, 1951] Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86.

[Lafferty et al., 2001] Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Lao et al., 2011] Lao, N., Mitchell, T., and Cohen, W. W. (2011). Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 529–539, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Le and Mikolov, 2014] Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org.

[LeCun and Bengio, 1998] LeCun, Y. and Bengio, Y. (1998). The handbook of brain theory and neural networks. chapter Convolutional Networks for Images, Speech, and Time Series, pages 255–258. MIT Press, Cambridge, MA, USA.

[Lee and Seung, 2000] Lee, D. D. and Seung, H. S. (2000). Algorithms for non-negative matrix factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, pages 535–541, Cambridge, MA, USA. MIT Press.

[Lee and Wang, 2016] Lee, J. and Wang, Y. (2016). Weighted rules under the stable model semantics. In *Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'16, pages 145–154. AAAI Press.

[Lee et al., 2018] Lee, N., Wu, C.-S., and Fung, P. (2018). Improving large-scale fact-checking using decomposable attention models and lexical tagging. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1133–1138, Brussels, Belgium. Association for Computational Linguistics.

[Lehmann et al., 2015] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2015).

DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2):167–195.

[Levy et al., 2014] Levy, R., Bilu, Y., Hershcovich, D., Aharoni, E., and Slonim, N. (2014). Context Dependent Claim Detection. pages pages 1489–1500.

[Levy et al., 2017] Levy, R., Gretz, S., Sznajder, B., Hummel, S., Aharonov, R., and Slonim, N. (2017). Unsupervised corpus–wide claim detection. *Proceedings of the 4th Workshop on Argument Mining*, pages 79–84.

[Liebeck et al., 2016] Liebeck, M., Esau, K., and Conrad, S. (2016). What to do with an airport? mining arguments in the german online participation project tempelhofer feld. In *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*, pages 144–153. Association for Computational Linguistics.

[Lim et al., 2016] Lim, W. Y., Lee, M. L., and Hsu, W. (2016). ClaimFinder: A Framework for Identifying Claims in Microblogs. *WWWWorkshop on Making Sense of Microposts*, 1691.

[Lim et al., 2017] Lim, W. Y., Lee, M. L., and Hsu, W. (2017). iFACT : An Interactive Framework to Assess Claims from Tweets. *CIKM*.

[Lippi and Torroni, 2016] Lippi, M. and Torroni, P. (2016). Argument Mining from Speech: Detecting Claims in Political Debates. *Proceedings of the AAAI*, pages 2979–2985.

[Liu et al., 2005] Liu, B., Hu, M., and Cheng, J. (2005). Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 342–351, New York, NY, USA. ACM.

[Liu and Lapata, 2018] Liu, Y. and Lapata, M. (2018). Learning structured text representations. *Transactions of the Association for Computational Linguistics*, 6:63–75.

[Lotan et al., 2013] Lotan, A., Stern, A., and Dagan, I. (2013). Truthteller: Annotating predicate truth. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 752–757, Atlanta, Georgia. Association for Computational Linguistics.

[Machado et al., 2019] Machado, C., Kira, B., and Howard, P. N. (2019). A Study of Misinformation in WhatsApp groups with a focus on the Brazilian Presidential Elections. *MisinfoWorkshop2019*, pages 1013–1019.

[Mahdisoltani et al., 2015] Mahdisoltani, F., Biega, J., and Suchanek, F. M. (2015). YAGO3: A knowledge base from multilingual Wikipedias. In *CIDR*.

[Mendes et al., 2011] Mendes, P. N., Jakob, M., García-Silva, A., and Bizer, C. (2011). Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 1–8, New York, NY, USA. ACM.

[Meng et al., 2018] Meng, Y., Shen, J., Zhang, C., and Han, J. (2018). Weakly-Supervised Neural Text Classification. *CIKM*, page 10.

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

[Mintz et al., 2009] Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Mukherjee and Weikum, 2015] Mukherjee, S. and Weikum, G. (2015). Leveraging joint interactions for credibility analysis in news communities. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM 2015)*.

[Nadeem et al., 2019] Nadeem, M., Fang, W., Xu, B., Mohtarami, M., and Glass, J. (2019). FAKTA : An Automatic End-to-End Fact Checking System. *NAACL*.

[Nakashole and Mitchell, 2014] Nakashole, N. and Mitchell, T. M. (2014). Language-Aware Truth Assessment of Fact Candidates. *ACL*, pages 1009–1019.

[Nguyen et al., 2019] Nguyen, T., Thanh Tam, N., Weidlich, M., Yin, H., Zheng, B., Quoc Viet Hung, N., and Stantic, B. (2019). User Guidance for Efficient Fact Checking. *VLDB*.

[Nie et al., 2019] Nie, Y., Chen, H., and Bansal, M. (2019). Combining Fact Extraction and Verification with Neural Semantic Matching Networks. *AAAI*.

[Ortona et al., 2017] Ortona, S., Meduri, V., and Papotti, P. (2017). Robust discovery of positive and negative rules in knowledge-bases. Technical Report EURECOM+5321, Eurecom.

[Pang and Lee, 2004] Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 271–278.

[Pang et al., 2002] Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Patwari et al., 2017] Patwari, A., Goldwasser, D., and Bagchi, S. (2017). Tathya: A multi-classifier system for detecting check-worthy statements in political debates. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 2259–2262, New York, NY, USA. ACM.

[Pellissier Tanon et al., 2018] Pellissier Tanon, T., Dias De Assuncao, M., Caron, E., and Suchanek, F. M. (2018). Demoing Platypus – A Multilingual Question Answering Platform for Wikidata. In *ESWC 2018 - Extended Semantic Web Conference*, pages 1–5, Heracklion, Greece.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *In EMNLP*.

[Peters et al., 2018] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proc. of NAACL*.

[Pomerleau and Rao, 2017] Pomerleau, D. and Rao, D. (2017). Fake News Challenge. fakenewschallenge.org.

[Popat et al., 2016] Popat, K., Mukherjee, S., Strötgen, J., and Weikum, G. (2016). Credibility assessment of textual claims on the web. In *Proceedings of the 25th ACM International*

on Conference on Information and Knowledge Management, CIKM '16, pages 2173–2178, New York, NY, USA. ACM.

[Popat et al., 2017] Popat, K., Mukherjee, S., Strötgen, J., and Weikum, G. (2017). Where the truth lies: Explaining the credibility of emerging claims on the web and social media. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, pages 1003–1012, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

[Prud'hommeaux and Seaborne, 2008] Prud'hommeaux, E. and Seaborne, A. (2008). SPARQL Query Language for RDF. W3C Recommendation. http://www.w3.org/TR/rdf-sparql-query/.

[Radim and Petr, 2010] Radim, Ř. and Petr, S. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages pages 45–50.

[Rajadesingan and Liu, 2014] Rajadesingan, A. and Liu, H. (2014). Identifying Users with Opposing Opinions in Twitter Debates. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction*, pages pages 153–160.

[Ratner et al., 2017] Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., and Ré, C. (2017). Snorkel: Rapid Training Data Creation with Weak Supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282.

[Ratner et al., 2016] Ratner, A., De Sa, C., Wu, S., Selsam, D., and Ré, C. (2016). Data Programming: Creating Large Training Sets, Quickly. *NIPS*, pages 1–27.

[Recasens et al., 2013] Recasens, M., Danescu-Niculescu-Mizil, C., and Jurafsky, D. (2013). Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1650–1659, Sofia, Bulgaria. Association for Computational Linguistics.

[Riloff and Wiebe, 2003] Riloff, E. and Wiebe, J. (2003). Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, pages 105–112, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Ristoski and Paulheim, 2016] Ristoski, P. and Paulheim, H. (2016). *RDF2Vec: RDF Graph Embeddings for Data Mining*, pages 498–514. Springer International Publishing.

[Rivas, 2019] Rivas, J. A. S. (2019). Examining the Roles of Automation , Crowds and Professionals Towards Sustainable Fact-checking. *MisinfoWorkshop2019*, pages 1001–1006.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA.

[Rus et al., 2013] Rus, V., Lintean, M. C., Banjade, R., Niraula, N. B., and Stefanescu, D. (2013). Semilar: The semantic similarity toolkit. In *ACL (Conference System Demonstrations)*, pages 163–168. The Association for Computer Linguistics.

[Saha et al., 2017] Saha, S., Pal, H., and Mausam (2017). Bootstrapping for Numerical Open IE. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 317–323.

[Schuster and Paliwal, 1997] Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

[Shi and Weninger, 2016] Shi, B. and Weninger, T. (2016). Fact checking in heterogeneous information networks. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, pages 101–102, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

[Sobhani et al., 2015] Sobhani, P., Inkpen, D., and Matwin, S. (2015). From Argumentation Mining to Stance Classification. *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 67–77.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

[Strötgen and Gertz, 2010] Strötgen, J. and Gertz, M. (2010). Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324, Uppsala, Sweden. Association for Computational Linguistics.

[Teh et al., 2004] Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2004). Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101.

[The World Wide Web Consortium (W3C), 2014] The World Wide Web Consortium (W3C) (2014). Best practices for publishing linked data. Available at: https://www.w3.org/TR/ld-bp/.

[Thomas et al., 2006] Thomas, M., Pang, B., and Lee, L. (2006). Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *EMNLP*, pages 327–335.

[Thorne and Vlachos, 2018] Thorne, J. and Vlachos, A. (2018). Automated Fact Checking: Task formulations, methods and future directions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3346–3359.

[Thorne et al., 2018] Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. (2018). FEVER: a large-scale dataset for Fact Extraction and VERification. *EMNLP*.

[Tschirschnitz et al., 2017] Tschirschnitz, F., Papenbrock, T., and Naumann, F. (2017). Detecting inclusion dependencies on very many tables. *ACM Trans. Database Syst.*, pages pages 18:1–18:29.

[W3C,] W3C. SPARQL protocol and RDF query language. http://www.w3.org/TR/rdf-sparql-query.

[W3C, 2004] W3C (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax.

[Wang, 2017] Wang, W. Y. (2017). "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computa-*

tional Linguistics (Volume 2: Short Papers), pages 422–426. Association for Computational Linguistics.

[Wang et al., 2018] Wang, X., Yu, C., Baumgartner, S., and Korn, F. (2018). Relevant Document Discovery for Fact-Checking Articles. *Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18*, pages 525–533.

[Wilson et al., 2005] Wilson, T., Wiebe, J., and Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Wu et al., 2014] Wu, Y., Agarwal, P. K., Li, C., Yang, J., and Yu, C. (2014). Toward computational fact-checking. *PVLDB*, 7(7):589–600.

[Xu et al., 2019] Xu, B., Mohtarami, M., and Glass, J. (2019). Adversarial domain adaptation for stance detection. *CoRR*, abs/1902.02401.

[Yang et al., 2016] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

[Zeng et al., 2014] Zeng, D., Liu, K., Lai, S., Zhou, G., Zhao, J., et al. (2014). Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

[Zhang, 2015] Zhang, C. (2015). *DeepDive: A Data Management System for Automatic Knowledge Base Construction*. PhD thesis.

[Zhi et al., 2017] Zhi, S., Sun, Y., Liu, J., Zhang, C., and Han, J. (2017). ClaimVerif: A Real-time Claim Verification System Using the Web and Fact Databases. *Proceedings of the 26th ACM International Conference on Information and Knowledge Management*, (3):2555–2558.

[Zuo et al., 2018] Zuo, C., Karakas, A. I., and Banerjee, R. (2018). A Hybrid Recognition System for Check-worthy Claims Using Heuristics and Supervised Learning. *CEUR Workshop Proceedings*.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Vers une vérification automatique des affirmations statistiques

Mots clés : Traitement Automatique du Langage naturel, Vérification des faits, RDF

Résumé : Toute personne ayant accès à Internet est potentiellement un producteur de contenu numérique. Bien que l'information soit facile d'accès, il est devenu de plus en plus difficile pour les consommateurs d'informations d'évaluer la crédibilité du contenu trouvé sur Internet. Un article d'actualité pourrait être partagé instantanément à des milliers de personnes qui peuvent ensuite le redistribuer, sans vérifier son contenu. En conséquence, la désinformation se déplace rapidement et peut avoir des conséquences dramatiques dans la vie réelle.

Il est difficile de vérifier la véracité des informations, même pour les professionnels comme les journalistes. Le journalisme de données et la vérification des faits sont des domaines d'intérêt croissant au sein de la communauté journalistique ainsi que dans l'audience en général, compte tenu de l'intérêt récent suscité par la désinformation, la manipulation par les médias et les efforts journalistiques visant à prévenir et dévoiler de telles tentatives.

Cette thèse a été développée dans le cadre d'une collaboration entre plusieurs laboratoires de recherche (Inria, LIMSI/CNRS and Université Paris Saclay, Université Rennes 1, Université Lyon 1) et Les Décodeurs, l'équipe de vérification des faits du journal Le Monde. La thèse proposait une approche de bout en bout pour la vérification automatisée des affirmations statistiques sur un sujet couvert par une base de données de référence.

Plus précisément, nous avons tout d'abord mis au point une méthode d'extraction des données ouvertes liées à partir des publications Web de l'INSEE (institut national de la statistique et des études économiques), le premier institut de statistiques français. Nous croyons que les données statistiques officielles constituent une très bonne information de base à utiliser pour évaluer la véracité d'une affirmation. Deuxièmement, nous avons développé un algorithme de recherche original qui, étant donné un ensemble de mots-clés tels que "taux de chômage France 2018", est capable de renvoyer les jeux de données (et, si possible, les valeurs exactes dans les jeux de données) jugés les plus pertinents par rapport aux mots-clés de l'utilisateur. Les défis consistent à quantifier la pertinence des jeux de données et à identifier correctement les extraits de données les plus pertinents. Troisièmement, nous avons développé une approche pour identifier automatiquement, dans un texte rédigé en français, les mentions d'entités statistiques, ainsi que les valeurs associées par le texte à ces entités, ainsi que d'autres termes de contexte (par exemple, heure ou lieu) attachés à l'affirmation statistique.

Ensemble, ces approches permettent un pipeline semi-automatisé de vérification des affirmations statistiques, tandis que les affirmations sont extraites automatiquement du texte et une requête est envoyée à notre algorithme d'extraction de données, qui renvoie les informations de référence les plus proches de la requête donnée. Un utilisateur, par exemple un journaliste, peut ensuite comparer les données à la valeur déclarée afin de les interpréter dans un travail de vérification des faits.

Dans un projet connexe à la vérification des faits, nous avons développé un système qui extrait des thèmes d'un corpus de texte (comprenant des tweets et des articles de nouvelles) et classe chaque texte en fonction de ces thèmes émergents. Il obtient des performances élevées en termes de précision de la classification. Ce système nous permet de comprendre les points de vue sur les mouvements sociaux actuels.

Nous envisageons de généraliser nos algorithmes à d'autres sources de données statistiques, par exemple la plateforme ouverte des données publiques françaises. Nous rechercherons également d'autres sources de données pouvant être utilisées pour la tâche de vérification des faits.

Cette thèse a été menée dans le cadre du projet ANR ContentCheck axé sur les modèles, les algorithmes et les outils pour le journalisme de données et la vérification des faits (http://contentcheck.inria.fr/).

Title : Toward Automatic Fact-Checking of Statistic Claims

Keywords : Natural Language Processing, Fact-checking, RDF

Abstract : Anyone with access to the Internet is potentially a producer of digital content. Although information is easy to access, it has become increasingly difficult for consumers of information to assess the credibility of content found on the Internet. A news article could be instantly shared to thousands of people who can then redistribute it, without checking its content. As a result, misinformation moves quickly and can have dramatic consequences in real life.
It is difficult to verify the veracity of the information, even for professionals such as journalists. Data journalism and fact-checking are areas of growing interest within the journalism community and also in the audience at large, given the recent interest in misinformation, manipulation through the media, and journalistic efforts to prevent and debunk such attempts.
This thesis has been developed within a collaboration between several research laboratories (Inria, LIMSI / CNRS and University Paris Saclay, University Rennes 1, University Lyon 1) and Les Décodeurs, the fact-checking team of Le Monde newspaper. The thesis proposed an end-to-end approach toward the automated fact-checking of statistic claims on a topic covered by a reference database.
Specifically, we have first devised an approach for extracting Linked Open Data from the Web publications of INSEE (National Institute of Statistics and Economic Studies), the leading French statistic institute. We believe that official statistical data is a very good background information to assess the truth of a claim. Second, we developed an original search algorithm which, given a set of keywords such as ”unemployment rate France 2018”, is capable of returning the datasets (and, if possible, the exact values within the datasets) deemed most relevant to the user keywords. Challenges include quantifying the relevance of the datasets and correctly identifying the most relevant data snippets. Third, we have developed an approach for automatically identifying, in a text written in French, mentions of statistic entities, together with the values associated by the text to these entities, and other context terms (e.g., time or place) attached to the statistic claim.
Together, these approaches enable a semi-automated statistic claim verification pipeline, whereas claims are extracted automatically from text and a query is sent to our data retrival algorithm, which returns the reference information closest to the given query. A human user, e.g., a journalist, can then compare the data to the claimed value in order to interpret it in a fact-checking work.
In a related fact-checking project, we developed a system that extracts topics from text (including tweets and news articles) and classifies each text according to these emerging topics. It gets high performance in terms of classification accuracy. This system allows us to understand the points of view on current social movements.
We plan to generalize our algorithms to other sources of statistical data, for example the open platform of French public data. We will also look for other sources of data that can be used for the fact-checking task.
This thesis has been carried on within the ANR ContentCheck project focused on models, algorithms and tools for data journalism and journalistic fact-checking (http://contentcheck.inria.fr/).

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l’Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France