UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE
ÉCOLE DOCTORALE MPSNI - Mathématiques Physique
Sciences du Numérique et de l'Ingénieur n°620

THÈSE

Pour obtenir le grade de DOCTEUR DE L'UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Discipline : INFORMATIQUE

Spécialité : INFORMATIQUE

Présentée et soutenue publiquement par

Mohamed Zohir KOUFI

Le 23 septembre 2024

Une Approche Collaborative pour la Classification des Sites Web

---

Thèse dirigée par **Zahia GUESSOUM, Amor KEZIOU**

JURY

M. Frederic BLANCHARD,	Maître de Conférences,	Université de Reims Champagne-Ardenne,	Invité
M. Mohand BOUGHANEM,	Professeur,	Université Paul Sabatier,	Rapporteur
Mme Zahia GUESSOUM,	Maître de Conférences HDR,	Université de Reims Champagne-Ardenne,	Directrice de thèse
M. Mathieu LAFOURCADE,	Maître de Conférences HDR,	Université de Montpellier,	Rapporteur
M. René MANDIAU,	Professeur,	Université Polytechnique Hauts-de-France,	Examinateur
M. Christophe MARSALA,	Professeur,	Sorbonne Université,	Examinateur
M. Smaine MAZOUZI,	Professeur,	Université 20 août 1955 Skikda,	Examinateur
M. Amor KEZIOU,	Maître de Conférences HDR,	Université de Reims Champagne-Ardenne,	Co-Directeur de thèse
Mme Itheri YAHIAOUI,	Maître de Conférences,	Université de Reims Champagne-Ardenne,	Co-Encadrante de thèse

Une Approche Collaborative pour la Classification des Sites Web

La croissance rapide de l'information numérique nécessite des systèmes de classification efficaces pour gérer et organiser le vaste répertoire de données en ligne. Cette thèse présente une approche collaborative pour la classification des sites web, tirant parti des méthodes traditionnelles et de l'apprentissage profond pour améliorer la précision et l'évolutivité. La recherche se concentre sur la base de données Olfeo, visant à affiner son processus de classification en utilisant des techniques de traitement du langage naturel (TAL) et d'apprentissage automatique.

L'étude commence par une revue de littérature complète, détaillant les techniques traditionnelles de représentation de texte ainsi que les nouvelles techniques basées sur l'apprentissage par des réseaux de neurones profonds. Par la suite, nous présentons une revue complète sur la classification des pages et sites web dans la littérature en commençant par les méthodes basées sur les approches d'apprentissage statistiques traditionnelles, les réseaux de neurones profonds, et les approches hybrides.

Elle introduit ensuite la méthodologie WeCA, en décrivant son cadre collaboratif et ses stratégies de combinaison de classifieurs tout en étudiant l'impact des donnés sur le mécanisme de la classification. La méthodologie est évaluée à travers des expériences approfondies sur des ensembles de données de pages web et de sites web, démontrant des améliorations significatives des performances de classification.

Une contribution clé de cette recherche est le développement d'une approche de fractionnement stratifié pondéré (WSSA) pour le découpage des données, améliorant la représentation des pages web dans les tâches de classification. De plus, l'application pratique de WeCA sur la grande base de données Olfeo démontre sa robustesse et son efficacité dans des scénarios réels. Les résultats mettent en évidence le potentiel de la combinaison de techniques traditionnelles et avancées pour atteindre une précision de classification et une adaptabilité supérieure.

Cette thèse contribue au domaine de la classification des sites web en présentant un modèle collaboratif et évolutif qui s'adapte à la nature changeante du contenu web. Les idées et méthodologies développées dans cette recherche fournissent une base pour les travaux futurs dans la catégorisation du contenu web, la recherche d'informations et les applications connexes.

Mots clés : Approches Collaboratives, Classification des Sites Web, TAL, Transformers, Apprentissage Profond, Apprentissage Automatique

A Collaborative Approach for Website Classification

The rapid growth of digital information necessitates efficient classification systems to manage and organize the vast repository of online data. This thesis presents a collaborative approach for website classification, utilizing both traditional and deep learning methods to enhance accuracy and scalability. The research focuses on the Olfeo database, aiming to refine its classification process using Natural Language Processing (NLP) and Machine Learning (ML) techniques. The proposed Website Classification Approach (WeCA) integrates text chunking and collaborative classifier combination strategies to address the challenges of long text classification and metadata impact.

The study begins with a comprehensive literature review, detailing traditional text representation techniques, statistical and neural network-based methods, and hybrid approaches. It then introduces the WeCA methodology, outlining its collaborative framework and classifier combination strategies. The methodology is evaluated through extensive experiments on web pages and website datasets, demonstrating significant improvements in classification performance.

A key contribution of this research is the development of a Weighted Stratified Split Approach (WSSA) for data splitting, enhancing the representation of web pages in classification tasks. Additionally, the practical application of WeCA on the Big Olfeo Database showcases its robustness and effectiveness in real-world scenarios. The findings highlight the potential of combining traditional and advanced techniques to achieve superior classification accuracy and adaptability.

This thesis contributes to the field of website classification by presenting a scalable, collaborative model that adapts to the evolving nature of web content. The insights and methodologies developed in this research provide a foundation for future work in web content categorization, information retrieval, and related applications.

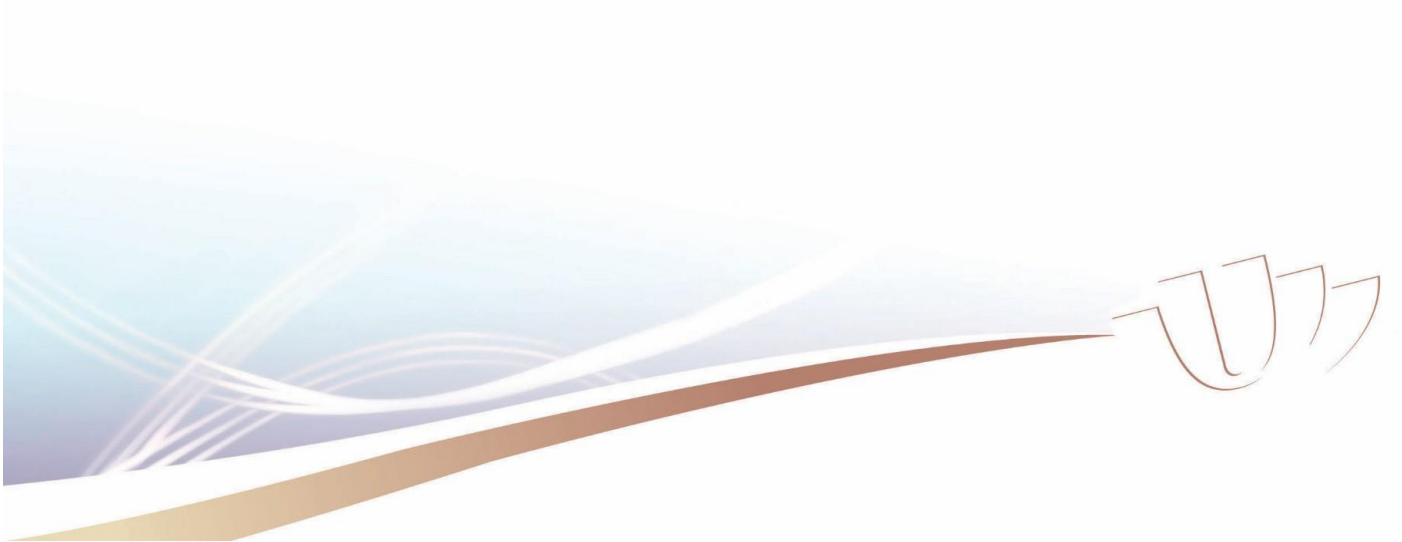
Keywords: Collaborative Approaches, Website Classification, NLP, Transformers, Deep Learning, Machine Learning

Discipline : INFORMATIQUE, Spécialité : INFORMATIQUE

Université de Reims Champagne-Ardenne

Lab-I* - UR 4474

UFR Sciences Exactes et Naturelles - Moulin de la Housse - BP 1039 - 51687 Reims CEDEX 2



Acknowledgments

This thesis grew from whispers in the dark,
Seeds of wonder planted deep in the heart.
Each word, each page, a tapestry spun,
From countless dreams, from battles won.

To my advisors and tutor, guides so wise,
Who taught me to seek, encouraged me to rise.
Your patient wisdom, profound and clear,
Illuminated paths through hope and fear.

Special thanks to Didier Schwab and Frederic Blanchard,
Whose kindness shone when times grew hard,
In desperate moments, your support sincere,
Your gentle guidance kept purpose near.

To those who doubted, who could not see,
Your skepticism sparked strength in me.
Each disbelief became a fire,
Fueling my journey, lifting me higher.

To friends and fellow researchers dear,
Whose laughter and insight brought clarity near,
Together we traversed the unknown,
Crafting wisdom from seeds we'd sown.

To my cherished siblings and nephews bright,
Who fill my days with joy and light,
Your love and laughter, pure and true,
Have colored this path with vibrant hue.

To my parents, pillars steadfast and strong,
Whose sacrifices resonate like a heartfelt song.
Through quiet endurance and selfless grace,
You've guided me gently, setting life's pace.
Your wisdom profound, your prayers sincere,

Nurtured resilience, calmed every fear.
In every step forward, each dream achieved,
It's your love and strength I've proudly received.

To seekers of truth who venture afar,
May these pages inspire wherever you are.
Pursue with courage, with heart sincere,
Knowledge awaits you, ever clear.

Above all, deepest thanks to God divine,
Who sees every hardship, every thought of mine.
Whose mercy, whose grace, forever pure,
Through faith alone, my trials endured.

And finally, a quiet thanks to me,
For daring to dream, to trust, to be.
In every challenge, every sleepless night,
Finding strength within, and holding tight.

With humble gratitude, this story I close,
Yet in wisdom and faith, forever it grows.

Mohamed Zohir KOUFI

"The man who can intelligently use the knowledge possessed by another is as much or more a man of education as the person who merely has the knowledge but does not know what to do with it."

— Napoleon Hill

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Problem Statement

The digital era has ushered in an unprecedented volume of data on the internet, presenting significant challenges in managing and organizing this vast repository of information. Efficient classification of this data is not just beneficial but essential for transforming raw data into actionable insights. Natural Language Processing (NLP) and Machine Learning (ML) are pivotal in this transformation, enabling sophisticated algorithms to parse, understand, and categorize content effectively.

The project's central aim is to harness NLP and ML to refine the classification process within the Olfeo database, thereby enhancing its ability to accurately categorize and utilize web-based information. This involves a multi-faceted approach: firstly, using NLP tools to automate the extraction of meaningful content from web pages; secondly, applying data mining techniques to analyze and derive patterns from this information; and thirdly, employing collective decision-making strategies to synthesize the insights from varied methodologies into a unified, optimal classification scheme.

This enhanced classification capability is expected to facilitate more nuanced and accurate data organization, which is crucial for a wide range of applications, from content filtering to information retrieval. As the volume and

complexity of online data continue to grow, the need for advanced, scalable classification systems becomes increasingly critical. The following sections will explore the state-of-the-art in text classification, highlighting recent research and developments in this rapidly evolving field.

1.2 Objectives

1.2.1 Text Classification

Text classification is a vital task in natural language processing, essential for assigning predefined categories to unstructured text. This process underpins numerous applications, from sentiment analysis in social media [1] to automated topic discovery in large datasets [2]. The main challenges here involve the development of robust feature extraction methods, accurate text representation, and the selection of optimal classifiers to manage varying data characteristics.

Traditional approaches like the Bag-of-Words (BOW) and Term Frequency-Inverse Document Frequency (TF-IDF) have long been the bedrock of text classification [3]. However, these methods often lead to significant semantic loss due to their inability to capture the deeper linguistic structures of text, resulting in sparse, high-dimensional representations that are challenging to manage. To mitigate these issues, dimensionality reduction techniques such as Principal Component Analysis (PCA) and Autoencoders have been utilized to compress the feature space while preserving essential information [4, 5].

The emergence of Deep Learning (DL) has dramatically shifted the landscape of text classification. Word Embeddings (WE), like Word2Vec [6], GloVe [7], and FastText [8], represent a paradigm shift by embedding words in continuous vector spaces where linguistic relationships are encoded as spatial relationships. These embeddings capture both implicit and explicit word

relationships, enhancing semantic understanding and reducing the need for complex feature engineering. Moreover, neural network architectures such as Convolutional Neural Networks (CNNs) [9], Recurrent Neural Networks (RNNs) [10], and Transformers [11] have enabled the exploration of deep patterns in text, improving classification accuracy. Ensemble methods like HDLTex [12], RMDL [13] and HTA-AGCRCNN [14] further refine this approach by combining multiple classifiers and learning strategies to enhance robustness and adaptability.

1.2.2 Towards a Collaborative Approach for Website Classification

The integration of traditional and deep learning methods in text classification suggests a promising avenue for synergistic improvements in accuracy. This thesis advocates for a hybrid, collaborative model that melds the strengths of both paradigms through consensus-based systems. The primary challenge in this integration is the effective aggregation of results from various classifiers into a unified decision-making framework. This necessitates a focused approach to the feature extraction phase, as the integrity and precision of feature representation critically impact classification outcomes.

This project will explore advanced hierarchical analysis techniques that balance granular detail with the overarching need to detect and model critical features effectively. This strategy addresses the challenges of irreversible text modeling, where early errors can significantly degrade overall classification performance. By promoting a collaborative architecture among different phases of the classification process, this thesis aims to develop a dynamic system that adapts to the evolving nature of web content, ensuring robust and precise classifications. This approach aligns with the principles outlined by , emphasizing the interdependence of feature detection and modeling in achieving accurate recognition.

1.3 Thesis Organization

This thesis is organized into four main chapters, each addressing different aspects of the website classification problem using a blend of traditional and deep learning methods.

Chapter 1 sets the stage by outlining the context and problem statement, emphasizing the importance of efficient data classification in the digital age. The objectives of the thesis are detailed, highlighting the challenges and advancements in text classification and the need for a collaborative approach to website classification.

The literature review (Chapter 2) comprehensively covers the evolution of text classification, beginning with traditional techniques like rule-based methods and statistical representations. It progresses to more advanced topics such as discrete vector space models, feature and dimensionality reduction, and conceptual embedding. The chapter also focuses into neural network-based representations, including static and dynamic word embeddings, and the latest developments in fine-tuning-based embeddings. The exploration of traditional machine learning methods like Naive Bayes, KNN, Decision Tree, and SVM is included, alongside a detailed examination of deep learning approaches encompassing MLPs, CNNs, RNNs, attention mechanisms, and transformers. Hybrid approaches are discussed to bridge the gap between various methodologies.

Chapter 3 introduces the Website Classification Approach (WeCA), focusing on collaborative classification techniques. It outlines the strategies for classifier combination, including hard and soft strategies, and the distinctions between adaptive and non-adaptive combiners. Advanced ensemble and hybrid approaches are explored, with a discussion on the merits and challenges of each. The chapter culminates with a presentation of the WeCA methodology, detailing web page and website classification strategies.

Chapter 4 addresses the role of text chunking in enhancing website clas-

sification. It discusses the impact of transformers on long texts and the approaches for data splitting, including the Basic Split (BS) and Weighted Stratified Split Approach (WSSA). The effectiveness of website classification through chunked web pages is evaluated, supported by experimental results and statistical analyses, including Cochran's Q test.

Chapter 5 presents the practical application of the WeCA methodology and its implementation on the Big Olfeo Database. It covers data and model orchestration, machine learning operations (MLOps), and provides a comprehensive comparative analysis of classifiers. The chapter also evaluates the performance of the WeCA methodology across different datasets, highlighting the implications and insights derived from these experiments.

The concluding chapter 6 summarizes the key contributions of the thesis and reflects on the broader perspectives of the research. It encapsulates the findings, discusses the implications, and suggests potential directions for future research in the field of website classification.

Chapter 2

Literature Review

2.1 Introduction

The exponential growth of digital content on the web has necessitated the development of sophisticated techniques for website and web page classification. Accurate classification is crucial for a wide range of applications, including improving search engine performance, enhancing user experience, enabling efficient content filtering, and supporting advanced data analytics. This chapter presents an exhaustive survey of the methodologies employed in website and web page classification, examining their evolution, strengths, and limitations in addressing the complexities of web-based information.

A major challenge in classifying websites and web pages arises from the diverse nature of their content, which includes text, images, videos, and other media formats. However, since the majority of this content is text-based, text classification techniques become crucial in addressing the overall challenges of website classification. This shift from the broader aspects of website classification to the specific focus on text analysis emphasizes the need to develop robust methods capable of handling the complexity and diversity inherent in web content.

In this chapter, we begin by reviewing traditional text representation techniques such as Bag-of-Words (BOW) and Term Frequency-Inverse Docu-

ment Frequency (TF-IDF), followed by modern deep learning approaches like Word2Vec, GloVe, and FastText to enhance semantic understanding. This foundational exploration sets the stage for examining current website and web page classification approaches, which are categorized into traditional, deep learning-based, and hybrid methods. Traditional approaches rely on predefined rules or heuristic algorithms for straightforward, albeit sometimes inflexible, solutions. Deep learning-based methods utilize neural networks such as CNNs and RNNs to dynamically learn and extract intricate patterns from web content. Hybrid approaches combine traditional and deep learning methods, employing ensemble techniques and multi-model systems to improve classification accuracy and robustness. This chapter provides a comprehensive overview of these methodologies, illustrating their evolution and convergence in addressing the challenges of web-based information classification.

This chapter is organized as follow: Section 1 discusses traditional statistical methods and their evolution and explores the influence of deep learning models on text representation and classification. Section 2 introduces and examines web page and website classification approaches using traditional machine learning methods. Section 3 focuses on classification approaches using neural network architectures and their contributions to the field. In Section 4, we focus on the hybrid classification approaches. Finally, we conclude the chapter with a general conclusion of the literature review.

2.2 Text Representation Techniques

In the domain of NLP, the representation of text plays a foundational role in determining the success of various tasks, from sentiment analysis to machine translation. Text representation techniques transform raw textual data into structured, numerical formats that can be ingested by ML models, bridging the gap between human language and computational algorithms. Over the years, several methods have been developed to achieve this, ranging from tra-

ditional Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) approaches to sophisticated embeddings and neural architectures. Each technique has its own strengths, limitations, and applicability, shaped by the evolving demands and challenges of the NLP domain. As many researchers have surveyed such techniques [15–22] we can observe that the predominantly work encompasses two pivotal steps. The preliminary step involves the conversion of input text (raw data) into a numerical format, typically represented as vectors or matrices. Subsequently, the design and implementation of models are orchestrated for the meticulous processing of the numerical data, aiming to achieve specified objectives or execute particular tasks. The discussed manuscript primarily focuses on the former step, elucidating how the evolution in text representation methodologies has fostered a significant advancement in NLP capabilities, transitioning from a rudimentary understanding of isolated textual components to a comprehensive analysis of textual data.

In their work, (Patil et al., 2023) [22] delineate the categorization of embedding learning methodologies into three distinct paradigms: rule-based, statistical, and neural-network-based approaches. In the nascent stages of NLP, exact matching techniques were predominantly employed, with Context Free Grammar (CFG) being instrumental for textual analysis. This phase was characterized by a substantial reliance on complex, rule-based systems, a characteristic prominently exhibited in the architecture of search engines of that era. With the progression of NLP, the paradigm shifted towards approximate matching, introducing a margin of error tolerance up to a defined threshold. However, this methodology posed inherent challenges due to the ambiguous nature of natural languages and the complexities associated with the development of exhaustive rule sets. These challenges inadvertently catalyzed the exploration of statistical methodologies.

The advent of statistical methodologies marked a significant shift in focus towards the analysis of word frequency. A spectrum of techniques emerged under this paradigm, inclusive of One Hot Encoding (OHE), Bag of Words (BoW), Term-Frequency (TF), and Inverse-Document-Frequency (IDF). In

comparison to their grammar-based antecedents, these statistical methodologies were relatively facile to implement and notably augmented the accuracy of models.

Nevertheless, the progression of NLP did not culminate here. Despite the advantages, statistical approaches were invariably susceptible to the curse of dimensionality. Additionally, the limitations in computational power adversely impacted their performance efficacy on large-scale datasets. Thus, the transition towards neural network-based approaches epitomizes the latest phase in the evolutionary trajectory of NLP, signifying a continual quest for enhanced text representation and analysis methodologies.

Furthermore, an integral yet often under-emphasized aspect of text representation is the process of text preprocessing, which serves as a precursor to the aforementioned steps. Text preprocessing encapsulates a range of techniques that aim to condition the raw text data into a cleaner, more standardized form conducive for subsequent representation and analysis. Techniques such as tokenization, stemming, lemmatization, stop-word removal, and normalization are instrumental in this phase. By efficiently curating and preparing the text data through these preprocessing measures, the downstream text representation methodologies can operate more effectively and derive more accurate and insightful interpretations of the textual data. The meticulous execution of text preprocessing techniques invariably plays a pivotal role in amplifying the efficacy and accuracy of NLP applications, underscoring its significance in the broader NLP workflow. In this section we will focus into the myriad text representation techniques following the same structure as presented in [22], explaining their mechanisms, nuances, and significance in powering today's cutting-edge NLP solutions.

2.2.1 Rule-based Representations

The inception of the NLP field primarily catered to search engines and question-answering systems relying heavily on pattern matching techniques,

utilizing sequences of characters, words, or parts of speech for matching patterns [23]. Due to computational constraints, this era saw the hard coding of regular expressions and complex logical rules. These patterns were pivotal in document search and ranking or generating scripted responses in chatbots. Despite being well-suited for context-free programming languages, employing Context-Free Grammar (CFG) for parsing natural language, which is context-sensitive, resulted in less flexible rules [24]. This approach was marked by rigidity and necessitated hard coding for matching or extracting text information, based on exact sequence or pattern matching.

The introduction of fuzzy expressions offered a respite by performing approximate matches, thus embodying uncertainties and vagueness inherent in natural languages [25]. Fuzzy-CFG notably captured ambiguity and impreciseness in natural language queries [26]. Moving past the traditional CFG approach, hard coding the semantics of specific words enhanced NLP accuracy, with applications like Keyword Search on Databases (KWS) translating natural language queries to SQL format [27–29]. However, this was a domain-specific solution with limited scope.

Notably, regular expressions considered word ordering and matching while ignoring word meaning, context, and frequency. Though capable of simple keyword searches and assisting in basic task executions, these CFG-based systems lacked depth in semantic understanding, proving inefficient for tasks like text summarization and machine translation. Consequently, researchers veered towards exploring statistical approaches to address these limitations.

2.2.2 Statistical Representations

Statistical representations in NLP fundamentally revolve around converting words into numerical vectors, thereby representing an entire corpus as a matrix comprised of these vectors. This conversion facilitates the reduction of documents of varying lengths to fixed-length numerical lists, thereby providing a structured, uniform format for analysis. One of the primary

advantages of such vector representations is the ability to employ linear algebraic operations for manipulating these vectors, and computing distances and similarities between them. This proves to be significantly beneficial as it alleviates the necessity for manual coding of regular expressions and nested conditional rules, which would otherwise be required to address a plethora of problems.

The statistical representations encompass three primary categories: discrete vector space, density vector space and continuous vector space.

2.2.3 Discrete Vector Space

In the discrete vector space (DVS), the text or corpus is subjected to a preprocessing phase initially. This phase includes the elimination of stopwords and the execution of stemming to ascertain the root or stem of each word. Such preprocessing techniques, like stemming and lemmatization, are instrumental in normalizing word endings, which ensures that variations of a word are collated under a singular token. Additionally, the structure of the matrices generated through these techniques is determined based on the entities represented by the rows and columns. For example, a matrix where rows signify documents and columns denote words or terms is termed as a document-term matrix. Through such structured representations, statistical methodologies offer a robust foundation for analyzing and deciphering patterns within textual data.

Bag of Words

The concept of "bag-of-words" (BoW) representation traces back to [30], serving as an alternative to the sparse one-hot vector representations. Under the BoW paradigm, a matrix representation of a corpus is constructed where each row signifies a distinct sentence, and each column represents a unique word from the vocabulary. The dimensions of this matrix, termed a 'document-term' matrix, are dictated by the number of sentences and the vocabulary size of the corpus, expressed as:

|number of sentences| x |V|.

As each sentence in the corpus is processed, the matrix is updated to reflect either the occurrence (or lack thereof) or the frequency of every word. The position of the sentence in the corpus and the dictionary position of the current word are denoted by the row and column indices in the matrix, respectively. Two variants of BoW emerge based on the nature of the values recorded in the matrix: Binary BoW, where entries are binary (0 or 1) indicating the absence or presence of a word, and non-binary BoW, where entries denote the frequency of each word occurrence. BoW has provided a foundation for handling textual data in a structured numerical format, setting a precedent for subsequent text representation methodologies.

Let's consider the example of a corpus with 3 documents: [D_1 : "The cat sat on the mat", D_2 : "The dog sat on the log", D_3 : "Cats and dogs are great pets"]. Its corresponding vocabulary is : ["and", "are", "cat", "cats", "dog", "dogs", "great", "log", "mat", "on", "pets", "sat", "the"].

Table 2.1: BoW example

Documents	and	are	cat	cats	dog	dogs	great	log	mat	on	pets	sat	the
D1	0	0	1	0	0	0	0	0	1	1	0	1	2
D2	0	0	0	0	1	0	0	1	0	1	0	1	2
D3	1	1	0	1	0	1	1	0	0	0	1	0	0

One-Hot Encoding

One-Hot Encoding (OHE) is a vector representation method often used in machine learning algorithms. In this method, each word in the vocabulary is represented by a vector of length equal to the size of the vocabulary (the total number of unique words in the corpus). The vector consists of all zeros except for a single one at the index corresponding to the position of the word in the vocabulary. In Table 2.2, we use the same last example for D1, D2 and D3.

N-grams

An n-gram is a contiguous sequence of n items from a given sample of text or speech. Firstly introduced in (Katz et al., 1987) [31], n-grams are widely used in NLP and computational linguistics. When processing text, an n-gram

Table 2.2: OHE example

Documents	and	are	cat	cats	dog	dogs	great	log	mat	on	pets	sat	the
D1	0	1	0	0	0	0	0	1	1	0	1	1	1
D2	0	0	0	0	1	0	0	1	0	1	0	1	1
D3	1	1	0	1	0	1	1	0	0	0	1	0	0

could be a sequence of words, letters, or phonemes. In [32–34], authors have shown the power of n-gram models in language modeling such as machine translation [34] and correcting spelling errors in context [32].

In the context of text mining and language modeling, n-grams offer a way to capture some amount of phrase and sentence level context. The parameter n is the order of the model. For instance, a unigram model (1-gram) only considers the probability of each individual word to appear in the text. A bigram model (2-gram) considers the probability of each word given the previous word, and a trigram model (3-gram) considers the probability of each word given the previous two words, and so on.

Mathematically, the probability of observing the i-th word w_i in the context of the previous $n-1$ words can be calculated using the formula:

$$P(w_i|w_{i-(n-1)}, w_{i-(n-2)}, \ldots, w_{i-1}) = \frac{P(w_{i-(n-1)}, w_{i-(n-2)}, \ldots, w_{i-1}, w_i)}{P(w_{i-(n-1)}, w_{i-(n-2)}, \ldots, w_{i-1})}. \tag{2.1}$$

Let's consider the sentence: "The cat sat on the mat." The 2-grams (bigrams) for this sentence would be: ['The cat', 'cat sat', 'sat on', 'on the', 'the mat'].

Table 2.3: Example Bigrams and Trigrams

Bigrams	**Trigrams**
"The cat"	"The cat sat"
"cat sat"	"cat sat on"
"sat on"	"sat on the"
"on the"	"on the mat"
"the mat"	-

Category-based Embedding

This methodology emphasizes the pre-identified categories across a dataset, contrasting with approaches where categories are dynamically recognized. The categories are encoded in columns, forming a 'document-category' matrix, where each column corresponds to a particular category. This setup is particularly effective when documents need to be sorted into predefined categories. Unlike OHE and BoW which assign a unique column for every word in the vocabulary, Categorical Bag-of-Embeddings (CBE) significantly trims down the dimensionality as the number of columns equates to the number of categories, which is typically much lower than the vocabulary size. For instance, in sentiment analysis, documents are mainly categorized as positive, neutral, or negative, thus requiring only three columns in the matrix. The sentiment of each word is determined through user-defined lexicons, updating the respective category count in the matrix for each document. This representation is described as dense, attributing to the fewer columns and mostly non-zero values in rows, thereby requiring lesser storage and accelerating the model training process due to reduced dimensions. This contrast with OHE and BoW showcases CBE's efficiency in both storage and computational speed when dealing with predetermined categorical document classification.

The creation of lexicons, crucial for sentiment analysis in a CBE approach, can be achieved through two primary methods. The heuristic method relies on human-assigned polarity scores to words, with prominent hand-crafted lexicons like Vader, AFINN, Emotion Lexicon, Bing Liu's Lexicon, MPQA Lexicon and General Inquirer [35–40] serving as notable examples. However, this method is labor-intensive, possibly incomplete in covering all sentiment-indicative words in a corpus, and poses challenges for languages unfamiliar to the user due to unknown word polarities. Alternatively, the non-sentiment-based method utilizes labeled input data to build a frequency dictionary, mapping words to their occurrence counts across different classes, e.g., positive or negative sentiments. This technique helps deduce sentiment based on word occurrence frequencies across classes, with an example being the word 'happy' mapped to its occurrence counts in positive and negative documents.

In the CBE representation, vectors comprise frequency or count values focusing on categorical (semantic) information, albeit at the expense of ignoring syntactic, contextual, and word-ordering information. This representation is more suited for applications dealing with short sentences, where word order changes don't dramatically alter meaning. It excels in category-based counting tasks like sentiment analysis of reviews, junk email detection, troll message identification, and sentiment or politeness assessment in chat messages, among others. Through this, the two lexicon composition approaches impact the efficiency and applicability of the CBE method in different text processing scenarios.

Term Frequency

The non-binary BoW methodology, while accounting for word frequency, overlooks the relative significance of a word within a document or across a corpus. A more nuanced understanding of word relevance, introduced by (Salton, 1968) [41] through Term Frequency (TF), is essential for yielding precise insights. This relevance can assist in identifying documents more pertinent to specific query keywords.

In the TF model, word frequency within a document is evaluated. However, solely relying on frequency can be misleading. For example, consider the word 'ocean' appearing 40 times in Document X and 800 times in Document Y. A simple frequency comparison would suggest higher relevance of the word 'ocean' in Document Y. Yet, if Document X consists of only 100 words while Document Y comprises 1 million words, 'ocean' evidently holds more weight in Document X.

To rectify this, Normalized Term Frequency (NTF) is employed, dividing the term frequency of a word by the total number of words in the document. This normalization delineates the importance of a word compared to others in the same document. Through NTF, Document X would attain a higher ranking for the query word 'ocean' as opposed to Document Y, due to its normalized term frequency.

In NTF, a document-term matrix is constructed with rows representing doc-

uments or sentences, and columns denoting unique words in the corpus. Unlike preceding models, this matrix incorporates normalized term-frequency values rather than binary or frequency count integers, thus reflecting the relative word importance within a document.

Upon projecting the NTF vectors into a vector space, the computation of similarity scores between a query and a document or between multiple documents is feasible by examining the cosine angle between the vectors. A cosine similarity value closer to 1 indicates a smaller angle between vectors, reflecting a similarity in word proportions used in the documents, regardless of document size. Conversely, a cosine similarity of 0 represents perpendicular vectors with no common words, albeit potentially discussing the same topic using different terminologies.

Table 2.4: NTF example

Documents	and	are	cat	cats	dog	dogs	great	log	mat	on	pets	sat	the
D1	0	0	0,38	0	0	0	0	0	0,38	0,38	0	0,38	0,65
D2	0	0	0	0	0,38	0	0	0,38	0	0,38	0	0,38	0,65
D3	0,41	0,41	0	0,41	0	0,41	0,41	0	0	0	0,41	0	0

While the NTF model enhances precision by accounting for word importance, it retains a limitation of treating words independently and necessitating exact word matches for clustering documents. Like its predecessors, NTF overlooks semantic relationships like synonyms, antonyms, and analogies. Despite these shortcomings, NTF presents a more refined frequency-based approach, elevating the understanding of word relevance within documents compared to the earlier BoW and OHE techniques.

TF-IDF

While Normalized Term Frequency (NTF) sheds light on the significance of a word within a particular document compared to other words in the same document, it falls short of evaluating a word's importance relative to the entirety of the documents in a corpus. This deficiency can cause inaccurate clustering of documents, especially when they contain commonly occurring words such as "is," "are," "in," or "on." NTF might inaccurately score such documents as highly similar due to the frequency of these common terms.

To better discern a word's uniqueness across a corpus, the Inverse Document Frequency (IDF) metric, introduced by [42], comes into play. IDF highlights the words that are unique to certain documents, representing distinctive topics or concepts, and thereby facilitates more accurate document differentiation and classification. Essentially, IDF lowers the similarity score when non-unique, widely spread words are considered in document clustering.

The formula for IDF is expressed as the ratio of the total number of documents in the corpus to the number of documents containing a specific term, where N represents the total documents, and df_t denotes the document frequency of term t_i. Contrary to NTF, which boosts similarity scores for shared words, IDF diminishes similarity scores if the shared words are commonplace across the corpus.

$$\text{IDF}(t, D) = \log\left(\frac{N}{df_t}\right) \tag{2.2}$$

Combining Term Frequency (TF) and IDF (TF*IDF) assigns a numeric value to a word's importance in a specific document, given its prevalence across the corpus. This amalgamation of TF and IDF has been recognized for its superior accuracy over previously discussed models in numerous studies. Nonetheless, TF-IDF isn't without flaws. The resultant matrices tend to be high-dimensional and sparse, which necessitated the adoption of dimensionality reduction techniques for better handling. Furthermore, TF-IDF encounters challenges in semantic understanding, often failing to recognize synonymy or differing spellings which result in dissimilar vector representations. Thus, advanced similarity measures like cosine [43] or Okapi BM25 [44] might still miss the mark in associating synonyms or distinguishing antonyms.

Table 2.5: TF-IDF example

Documents	**and**	**are**	**cat**	**cats**	**dog**	**dogs**	**great**	**log**	**mat**	**on**	**pets**	**sat**	**the**
D1	0	0	0,46	0	0	0	0	0	0,46	0,35	0	0,35	0,59
D2	0	0	0	0	0,46	0	0	0,46	0	0,35	0	0,35	0,59
D3	0,41	0,41	0	0,41	0	0,41	0,41	0	0	0	0,41	0	0

In conclusion, while TF-IDF significantly improves document clustering and word importance assessment over earlier methods, it still exhibits limitations in semantic comprehension and requires dimensionality reduction strategies to manage high-dimensional data effectively.

The Hyperspace Analogue to Language

The Hyperspace Analogue to Language (HAL) introduced in [45], brought forth an automated method to explore word relationships using a co-occurrence matrix. This matrix forms coordinates in a high-dimensional semantic space, facilitating analysis of word interconnections, as discussed in [46]. Unlike previous attempts, where semantic spaces required manual definition of axes and meanings—an error-prone approach due to judgement bias, HAL automates this process using lexical co-occurrence techniques.

In HAL, the co-occurrence matrix is a square matrix, with its dimensions equivalent to the vocabulary size, denoted by |V|. A context window slides across text, capturing a few words at a time, and for each word (represented by a row), every word preceding it within this window is deemed as co-occurring. The values in the corresponding columns are increased by 1 for every such co-occurrence. Essentially, this matrix documents the frequency of co-occurrences between word pairs.

Take for instance a simple sentence: “The cat chased the mouse.” With a window size of 4, a co-occurrence matrix would be generated, documenting how often words appear in the vicinity of others. Once the matrix is constructed, distance metrics are employed to measure semantic similarity between word pairs, which in turn helps in categorizing and classifying documents.

For example, the semantic similarity between the vectors for "cat" and "mouse" would likely be higher compared to that between "cat" and "chased" because the probability of "cat" and "mouse" co-occurring is higher. When plotted in a multi-dimensional space, vectors close to each other form a meaningful cluster, indicating a strong semantic relationship.

HAL’s approach lies in representing each word vector based on the contex-

tual words surrounding it, indirectly inferring the word's meaning from its context. However, a limitation of the HAL model is that common words can disproportionately influence the similarity measure. Frequent words, although might have less semantic significance, could heavily skew the similarity scores between word vectors.

For instance, if the words "book" and "movie" frequently share similar context words, their vectors would be projected closer to each other in the semantic space, suggesting a high similarity score, despite their different meanings. This shows that while HAL offers a robust automatic approach to uncovering word relationships, it can be misled by frequently occurring words, thus requiring further refinement to accurately represent semantic relatedness between words.

2.2.4 Features and Dimensionality Reduction

Discrete space techniques, numerically represent a corpus using vectors and matrices. While simple in execution, they often face issues of high dimensionality and data sparsity, making model training time-consuming. To address these challenges, dimensionality reduction techniques, categorized into feature selection and feature transformation, were employed to represent information in a more condensed manner.

Feature Selection Techniques

Feature selection is a technique where only a subset of the original dimensions is chosen as features. Within this domain, there are four primary methods: Document Frequency (DF), Term-Frequency Variance (TFV), Mean TF-IDF (MTI), and Information Gain (IG).

Document Frequency (DF) denotes the count of documents where a term appears at least once. Terms are sorted based on their DF values, and those with values above a certain threshold are selected, deeming the rest as non-informative for category prediction. In contrast, the TFV method involves ranking terms based on the variance of their frequency, capturing the term's

quality [47]. Only the top-k terms with the highest variance values are selected as features. Additionally, χ^2 (Chi-squared) [48] is a feature selection method that measures the dependence between the term and the class label. It assesses the probability that the observed distribution of the term across various class labels occurs by chance. A high score indicates that the term likely depends on the class and might be a good discriminative feature.

Meanwhile, the MTI approach determines the importance of a term by calculating the MTI value for each term across all documents [49]. Terms are then ranked by these values, and the top-k ones that meet the threshold criteria are chosen. Lastly, Information Gain (IG) assesses how the presence or absence of a term influences the obtained information for category prediction [50], determining which feature is most beneficial in distinguishing between classes.

In essence, these feature selection methods involve ranking terms based on various criteria and then selecting a subset that meets specific thresholds, ensuring the most relevant features are included.

Feature Transformation

Feature transformation reduces the dimensionality of data by converting a high-dimensional space into one with fewer dimensions. Each dimension in this reduced space is a linear or non-linear combination of the original dimensions. Popular linear transformation methods include latent semantic indexing, latent Dirichlet allocation, random indexing or projection, and independent component analysis.

Latent Semantic Indexing (LSI) [51], utilizes Singular Value Decomposition (SVD) to group co-occurring words under common topic vectors, enabling documents with similar terms to have comparable representations in a latent semantic space. This approach assigns greater weight to words that strongly represent a topic, compressing the vocabulary into fewer topic-representing columns and reducing the overall number of features. LSI is particularly effective for smaller documents and can handle synonyms by identifying high-order semantic structures, making it valuable for automatic indexing and

improving access to textual information in information retrieval [52]. However, LSI has limitations with polysemy, which Probabilistic Latent Semantic Indexing (PLSI) addresses in [53].

PLSI uses a generative data model to manage polysemous words through a mixture model representing "topics". Unlike LSI, PLSI employs a latent variable model that allows documents to exhibit multiple topics, enhancing the granularity and accuracy of topic representation. This model makes PLSI a more sophisticated approach in handling the complexities of language, enabling improvements over LSI in perplexity results for various text and linguistic data collections [53].

Building on these ideas, Latent Dirichlet Allocation (LDA) [2] further extends the concept by allowing documents to show multiple topics to varying degrees. LDA treats topic mixtures as hidden random variables, which enables the model to generalize to new documents without a linear increase in parameters. LDA's nonlinear approach provides a more accurate allocation of words to topics than LSI but requires more training time, making it suitable for specific applications like document summarization.

Transitioning from these probabilistic models, Independent Component Analysis (ICA) [54] explores data transformation from a different angle. Related to blind source separation, ICA analyzes linear mixtures of independent components to produce statistically independent results. Unlike LSI, which seeks orthogonal combinations, ICA aims for components that are as independent as possible. This method often uses Principal Component Analysis (PCA) as a preprocessing step and is applied in various fields, including audio and image processing [55].

Finally, Random Indexing (RI), based on the Johnson-Lindenstrauss lemma [1], preserves distances between points when projecting them into a randomly selected subspace. RI's incremental and scalable nature allows for dimensionality reduction without the need for initial sampling of the entire dataset, making it computationally less expensive than SVD. RI maintains constant

[1] https://dash.harvard.edu/handle/1/17369243 (accessed on 01/09/2023)

dimensionality and provides results comparable to PCA, facilitating efficient processing and analysis.

Together, these methods represent a spectrum of approaches for feature transformation, each with its strengths and nuances, offering robust solutions for managing and interpreting complex data in various applications.

2.2.5 Conceptual Embedding

Unlike the conventional BoW method, which relies on word occurrences for text categorization, Explicit Semantic Analysis (ESA) enhances the approach by incorporating common sense and broader contextual knowledge. Proposed by [56], ESA utilizes concepts from Wikipedia to enrich document representation. It represents texts as a weighted combination of predefined natural concepts, effectively mapping text onto a semantic space.

For instance, imagine how ESA would interpret a text about solar technology. While traditional BoW might focus solely on word frequencies, ESA would go further, connecting phrases or words to relevant Wikipedia concepts like "Photovoltaics," "Solar Energy," or "Renewable Resources," creating a more detailed representation. ESA's backbone is a semantic interpreter, which transitions the text into a sequence of Wikipedia concepts, weighted by relevance. The resultant "interpretation vectors" embody the text's affinity with these Wikipedia concepts. For instance, in the solar technology text, a heavy weight might be assigned to the "Solar Energy" concept, reflecting its centrality to the text. This model excels in identifying semantic relatedness among texts. By comparing their respective interpretation vectors through distance metrics like cosine similarity, the semantic closeness of different texts can be evaluated.

At the core of this process is an inverted index, keeping track of the concepts associated with each word. As ESA sifts through the text, it aggregates the relevant concepts from this index into a weighted vector, with each entry denoting a concept's relevance to the text. Contrary to frequency-based

methods, ESA relies on Wikipedia's conceptual richness, providing a solid understanding rather than mere word counts. This diverges from Latent Semantic Indexing (LSI) as well, where the emphasis is on "latent concepts", generated from tf-idf features, lacking the grounded human cognition inherent in ESA's approach.

Through contextual analysis, ESA maps documents onto Wikipedia or ontology concepts to generate features. This contextual approach empowers ESA to tackle Natural Language Processing challenges like synonyms and polysemy, delivering a semantic depth that's instrumental in fostering a more intuitive, human-like understanding of text.

2.2.6 Density-Based Distributed Embeddings

Traditional word embedding methods often utilize either discrete or continuous vector spaces to represent words. However, a different approach explores the density-based distributed embeddings, where words are characterized through Gaussian distributions within a latent embedding space. In this setup, each word's distribution, denoted by a mean and covariance matrix learned from the data, encapsulates its linguistic properties.

Let's focus into this with an example. Rather than showing the word "apple" as a single point in a vector space, it's portrayed as a Gaussian distribution across a hidden space. This doesn't just depict "apple" but includes associated concepts like "fruit", enabling a more nuanced grasp of semantic connections.

The beauty of Gaussian embeddings, as illustrated in [57], lies in its capacity to provide a richer geometrical representation, allowing for a direct portrayal of probability mass and uncertainty which is crucial for semantic understanding.

Yet, a hurdle emerges when dealing with polysemous words—words with multiple meanings. The unimodal nature of Gaussian distribution struggles to represent such words accurately. For instance, the word "bank" can refer

to a financial institution or the side of a river, but a single Gaussian distribution may create a generalized or biased representation leaning towards one meaning over the other.

An ingenious solution is introduced in [58], proposing a multimodal distribution approach, where each word is modeled using a mixture of gaussians. In this model, each Gaussian component signifies a distinct meaning of the word. Now, our word "bank" can have two Gaussian components, one leaning towards the financial context while the other towards a riverbank context.

This multimodal model doesn't just stop at a fixed number of Gaussian mixtures. It also offers a training methodology to learn the parameters of these mixtures, optimizing the representation of each word with multiple meanings. Each Gaussian component is visualized as an ellipsoid, with its center specified by the mean vector representing one distinct meaning of the word, and the contour surface (described by the covariance matrix) reflecting nuances in meaning and associated uncertainty.

For instance, in a graphical illustration, the word "plant" might have one Gaussian component aligned with "flora" and "tree", while another component aligns with "factory" and "industry", reflecting its distinct meanings and related contexts.

This method steps away from point embeddings, advancing towards capturing the richness and multiplicity of semantic relationships, essential for predictive tasks in various applications. Through this, a more holistic and nuanced semantic understanding is facilitated, paving the way for more accurate and contextually rich word representations in natural language processing tasks.

2.2.7 Neural Network-based representations

The embedding methodologies discussed in prior sections predominantly adhere to rule-based or statistical paradigms. A departure from this traditional stance is observed in neural network-based methods which automate

the extraction of features, thereby encapsulating the syntactic and semantic nuances inherent to language. This automation significantly mitigates the manual rigors entailed in feature engineering. Customarily, these neural models use tasks related to Language Modeling (LM) or Machine Translation (MT) to engineer the embeddings. The resultant embeddings are real-valued vectors that embody the semantic dimensions of words. The computational intricacy of such models is inherently dictated by the quantity of parameters necessitated for training.

The delineation of neural network-based embedding methodologies is broadly categorized into two distinct paradigms: feature-based and fine-tuning-based embeddings. In the feature-based paradigm, a pre-existing network is employed to yield language representations of varying granularities, encompassing word, phrase, or sentence embeddings. This paradigm further extends into two subclasses, namely static and dynamic embeddings. Static embeddings, characterized by their non-contextual nature, maintain a static representation of a word irrespective of its contextual utilization. Such representations are traditionally derived employing networks of lesser complexity. Conversely, dynamic embeddings exhibit a fluid representation of words, contingent on the contextual ambiance, thereby proficiently addressing the aspect of polysemy, which pertains to the multiple meanings a word may possess.

On the other hand, the fine-tuning-based paradigm initiates with a pre-training phase on a language modeling objective executed in a self-supervised manner. This phase is subsequently followed by a fine-tuning procedure on downstream tasks such as classification, Named Entity Recognition (NER), or Question Answering (QA) utilizing supervised data. This sequential procedure facilitates the model in customizing the generic language representations to cater to specific task requisites, thereby potentially ameliorating the performance on those specified tasks.

The dichotomy into feature-based and fine-tuning-based methodologies elucidates the versatility and adaptability ingrained in neural network-based

embeddings for various linguistic representation chores. Each methodology, with its unique operational modus operandi, fosters a more profound comprehension and superior representation of language, thus significantly contributing towards effectively addressing a spectrum of challenges encountered in NLP endeavors.

2.2.8 Static Word Embeddings

Word2Vec

Word2Vec [59] is a popular method for learning dense vector representations of words, often called "word embeddings". These embeddings are learned based on the context in which words appear in the text, capturing semantic and syntactic relationships between words. Word2Vec uses a shallow neural network model to learn these word representations. The model is trained to reconstruct the linguistic context of words. It comes in two flavors: the Continuous Bag of Words (CBOW) model and the Skip-Gram model.

In the CBOW model, the distributed representations of context (or surrounding) words are combined to predict the word in the middle. While in the Skip-Gram model, the distributed representation of the input (or center) word is used to predict the context words.

The Skip-Gram model of Word2Vec is trained to find word representations that are useful for predicting the surrounding words in a sentence or a document. More formally, given a sequence of training words $w_1, w_2, ..., w_T$, the objective of the Skip-Gram model is to maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t), \tag{2.3}$$

where c is the size of the training context. The basic Skip-Gram formulation defines $p(w_{t+j}|w_t)$ using the softmax function :

$$p(w_O|w_I) = \frac{\exp({v'_{w_O}}^T v_{w_I})}{\sum_{w=1}^{W}\exp({v'_w}^T v_{w_I})}, \tag{2.4}$$

where v_w and v'_w are the "input" and "output" vector representations of w, and W is the number of words in the vocabulary.

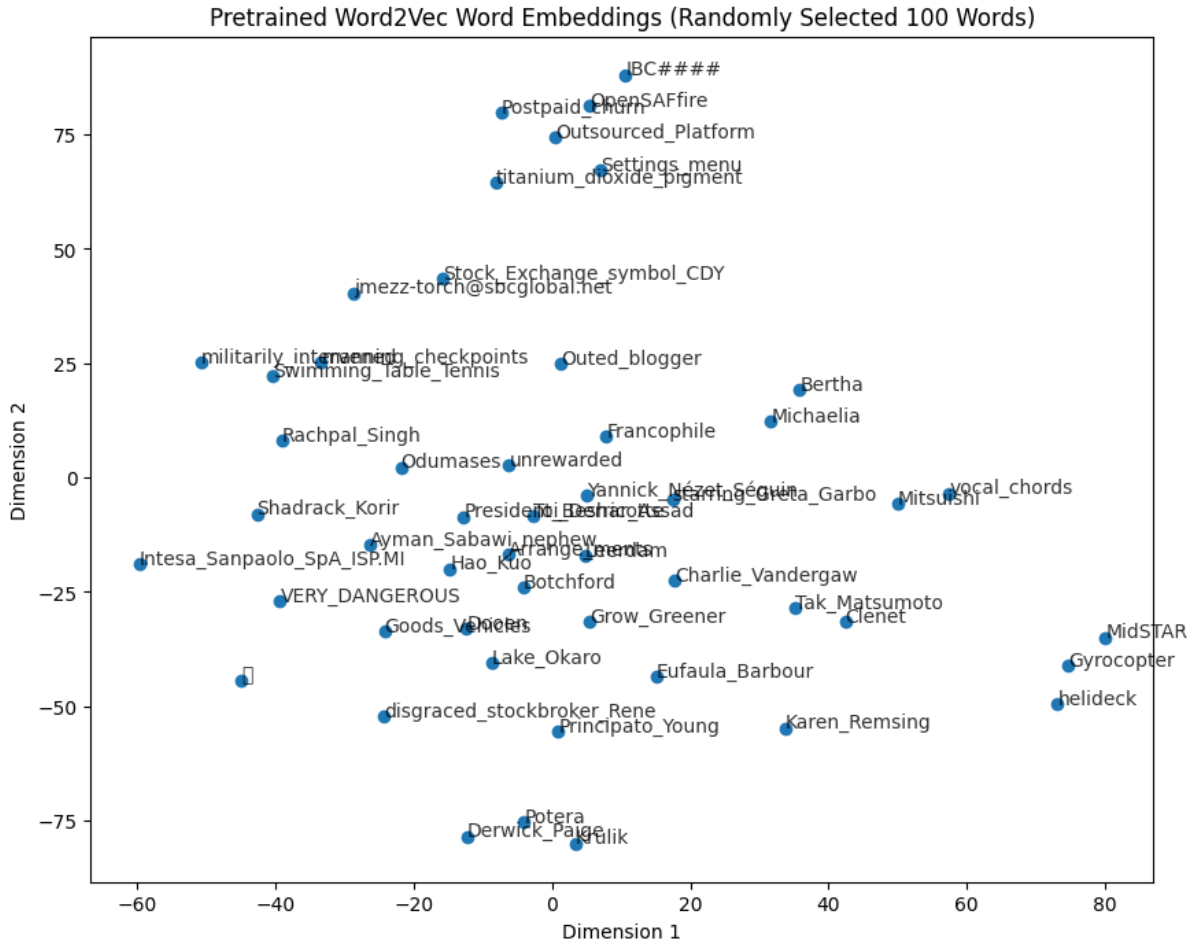


Figure 2.1: Pretrained Word2Vec model example

Doc2Vec

The Doc2Vec model [6], also known as Paragraph Vector, is an extension of the Word2Vec model that learns distributed representations of documents. It allows us to represent variable-length pieces of text, such as sentences, paragraphs, or entire documents, as fixed-length vectors. The model aims to capture the semantic meaning and context of the documents.

The main idea behind Doc2Vec is to learn document embeddings by predicting words within a document using continuous vector representations. Two main architectures are commonly used in the Doc2Vec model: Distributed Memory (DM) and Distributed Bag of Words (DBOW).

Distributed Memory (DM) Architecture: In the DM architecture, the

model learns to predict a target word from a given context, considering both the word vectors and a unique document vector. The document vector serves as a memory that helps incorporate document-level information into the prediction process.

Mathematically, let's consider a document d with a sequence of words $w_1, w_2, ..., w_T$. The objective of the DM architecture is to maximize the average log probability of predicting the target word w_t given its context words $w_{t-k}, ..., w_{t-1}, w_{t+1}, ..., w_{t+k}$ and the document vector d:

$$\frac{1}{T}\sum_{t=k}^{T-k}\log p(w_t|w_{t-k}, ..., w_{t-1}, w_{t+1}, ..., w_{t+k}, d). \tag{2.5}$$

The probability $p(w_t|w_{t-k}, ..., w_{t-1}, w_{t+1}, ..., w_{t+k}, d)$ is computed using the softmax function :

$$p(w_t|w_{t-k}, ..., w_{t-1}, w_{t+1}, ..., w_{t+k}, d) = \frac{\exp(v'_{w_t} \cdot v_d)}{\sum_{j=1}^{W}\exp(v'_{w_j} \cdot v_d)}, \tag{2.6}$$

where v'_{w_t} represents the vector representation of word w_t, v_d represents the document vector, and W is the total number of words in the vocabulary.

Distributed Bag of Words (DBOW) Architecture: In the DBOW architecture, the model treats each document as an unordered bag of words and learns to predict randomly sampled words from the document using only the document vector. Given a document d with a sequence of words $w_1, w_2, ..., w_T$, the objective of the DBOW architecture is to maximize the average log probability of predicting a word w_t sampled from the document, given the document vector d:

$$\frac{1}{T}\sum_{t=1}^{T}\log p(w_t|d). \tag{2.7}$$

Similar to the DM architecture, the probability $p(w_t|d)$ is computed using the softmax function.

The model is trained using techniques such as negative sampling or hierarchical softmax to optimize the objective function. After training, the document vectors can be used as fixed-length representations for various document-level tasks, including document classification, clustering, and similarity analysis.

GloVe

The GloVe model [7] is a popular unsupervised learning algorithm for obtaining dense word representations, often referred to as word embeddings. It combines the advantages of both global matrix factorization methods and local context window-based methods to capture semantic and syntactic relationships between words. The key idea behind the GloVe model is that word co-occurrence statistics can be used to learn word vectors. The model aims to learn word embeddings that are capable of capturing the ratios of word co-occurrences across the entire corpus. These ratios provide meaningful information about word semantics.

Let's consider a co-occurrence matrix X, where each entry X_{ij} represents the number of times word j appears in the context of word i in the corpus. The GloVe model seeks to learn word vectors $\mathbf{w}_i$ and $\mathbf{w}_j$ such that their dot product, along with the biases b_i and b_j, captures the logarithm of the word co-occurrence ratio :

$$\mathbf{w}_i^T \mathbf{w}_j + b_i + b_j = \log(X_{ij}). \tag{2.8}$$

To prevent over-fitting and provide better generalization, the model introduces a weighting term that assigns more importance to rare word co-occurrences :

$$\mathbf{w}_i^T \mathbf{w}_j + b_i + b_j = \log(X_{ij}) - \log(X_i), \tag{2.9}$$

where X_i represents the total number of times word i appears in the corpus.

The model defines a cost function to measure the discrepancy between the left-hand side and the right-hand side of the equation. It minimizes the sum of squared errors over all word pairs:

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij}) \left(\mathbf{w}_i^T \mathbf{w}_j + b_i + b_j - \log(X_{ij}) + \log(X_i) \right)^2, \tag{2.10}$$

where V represents the size of the vocabulary, and $f(X_{ij})$ is a weighting function that allows for finer control over the importance of different word co-occurrence pairs.

The GloVe model is trained using gradient descent optimization to minimize the cost function. The word vectors $\mathbf{w}_i$ and biases b_i are updated iteratively to find the optimal values that capture the word co-occurrence ratios.

The GloVe model offers several advantages. It is able to capture both global semantic relationships and local context-based information. It is computationally efficient compared to some other models. It also provides meaningful representations even for rare words and out-of-vocabulary words. However, the GloVe model has a few limitations. It requires a large corpus to effectively capture the word co-occurrence statistics. It may not fully capture fine-grained semantic relationships between words. It also does not consider the order of words within the context window.

FastText

The FastText model [60] is an extension of the Word2Vec model that is designed to capture subword information in addition to word-level information. It allows for better representation of rare words and out-of-vocabulary words by utilizing character n-grams. The main idea behind FastText is to represent each word as a bag of character n-grams, which are contiguous sequences of characters of length n. By considering character n-grams, the model can capture morphological and semantic information from the subwords.

Given a word w, the FastText model represents it as a combination of its character n-grams. Let $G(w)$ denote the set of character n-grams of word w. The word vector representation of w, denoted as v_w, is computed as the sum of the vector representations of its character n-grams :

$$v_w = \sum_{g \in G(w)} z_g, \tag{2.11}$$

where z_g is the vector representation of character n-gram g. Each character n-gram has an associated vector representation, which is learned during the training process.

The FastText model employs a skip-gram approach similar to Word2Vec for learning the vector representations. The objective is to maximize the average log probability of predicting the surrounding words given a target

word. However, in FastText, the target and context words are represented by their character n-grams. Mathematically, let w be the target word and c be a context word. The objective function of FastText is defined as :

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(c|w_t), \tag{2.12}$$

where T is the total number of target-context pairs in the training corpus and $p(c|w_t)$ is the probability of predicting context word c given the target word w_t. The context word c is represented by its character n-grams. To compute the probability $p(c|w_t)$, FastText uses the softmax function, similar to Word2Vec. The softmax probability is calculated as :

$$p(c|w_t) = \frac{\exp(v_c \cdot v_{w_t})}{\sum_{c'\in C}\exp(v_{c'} \cdot v_{w_t})}, \tag{2.13}$$

where v_c is the vector representation of the context word c, and C is the set of all possible context words.

The FastText model is trained using stochastic gradient descent (SGD) to minimize the negative log likelihood of the target-context pairs. After training, the learned vector representations can be used for various natural language processing tasks such as text classification, information retrieval, and word similarity.

2.2.9 Dynamic Word Embeddings

The practice of transferring information through predefined word vectors such as Word2Vec and GloVe, as opposed to random initialization, has demonstrated improved performance across various tasks. Despite this advancement, a notable limitation of such representations is their context-independent nature, rendering them inadequate for disambiguating word senses based on surrounding context. This inadequacy is particularly manifest in scenarios where an ambiguous word assumes multiple, potentially unrelated meanings depending on its context. Therefore, the need for context-sensitive representations is imperative to bolster transfer learning, especially

in NLP tasks like named entity recognition, word sense disambiguation, and co-reference resolution where contextual understanding is paramount.

Addressing this limitation, the emergence of contextualized word embeddings has marked a significant stride. Unlike static embeddings, contextualized embeddings are dynamic and adapt based on the contextual environment of words. The pioneering models spearheading this advancement include CoVe, ELMo, GPT, and BERT [61–64]. These models excel in generating contextual embeddings that morph in alignment with the context, thus offering a more nuanced representation of word semantics.

The inception of these embeddings is primarily through pre-trained Language Modeling (LM) tasks, which learn to predict words based on their contextual milieu using substantial text datasets. The training of the LM on a general-domain corpus facilitates the capture of generic linguistic features across different layers. This pre-training phase sets the foundation for deriving context-rich embeddings, which are instrumental in comprehending the semantics of words as dictated by their context.

The narrative now transitions towards a discussion on prominent contextual embeddings engineered through various neural network architectures. These architectures, underpinned by the principle of contextual understanding, epitomize the evolving trajectory of embedding models geared towards enhancing transfer learning in NLP. Through these advancements, the NLP realm is better positioned to tackle tasks necessitating a deep-seated understanding of contextual interplays among words, thus broadening the horizon of what can be achieved in language processing tasks.

Context2Vec

The Context2Vec (C2V) model, as proposed in reference [65], is an unsupervised model adept at learning generic, task-independent representations of a wide sentential context surrounding a target word, using a bidirectional Long Short-Term Memory (LSTM) architecture. Unlike traditional models that may struggle with variable-length contexts, C2V can encapsulate these contexts within a fixed-size vector. It draws inspiration from the Continuous

Bag Of Words (CBOW) architecture but elevates its capability by replacing the simplistic neural model with a robust parametric model furnished by bidirectional LSTM.

In this architecture, two LSTMs operate in tandem where one processes the input from left to right and the other from right to left. These networks function independently with distinct parameters for both left-to-right and right-to-left word embeddings. Following this processing, the outputs from both LSTMs are amalgamated through concatenation.

A significant divergence from CBOW is observed in C2V's ability to consider the entire sentence to derive the sentential context, superseding the limited context window size in CBOW. This expanded contextual comprehension is instrumental in capturing pertinent information that might be remotely located from the target word. Consequently, target words that share similar sentential contexts are observed to possess similar embeddings.

The efficacy of C2V in preserving crucial linguistic attributes like part-of-speech and tense information, courtesy of the extensive sentential context, is demonstrated in [65]. The experimental validation of C2V was conducted across three distinct tasks: Sentence Completion, Lexical Substitution Task (LST), and Word Sense Disambiguation (WSD). Remarkably, C2V outperformed the Average-of-Word-Embeddings (AWE) model across all evaluated benchmarks, showcasing its superior capability in rendering rich contextual representations conducive for various linguistic tasks.

Context Vectors

The Context Vectors (CoVe) approach [61], utilizes a neural machine translation (NMT) encoder to compute contextualized representations. CoVe extracts its vectors from the deep LSTM encoder of an attention-based sequence-to-sequence model initially trained for machine translation tasks. Post training, this LSTM encoder gets repurposed for various Natural Language Processing (NLP) tasks, exhibiting a notable transfer of learning. The study underscores the potential of machine translation data, analogous to the CNN's ImageNet in computer vision, as a foundational source for developing

reusable models.

This method draws a parallel between the pairing of MT and LSTM, and the well-known ImageNet and CNN pairing in the domain of computer vision. An evident improvement in the performance of downstream NLP tasks is observed when CoVe vectors are appended to the word vectors of the model, compared to solely utilizing pre-trained word vectors from baseline models.

The fixed-length representations procured from the NMT encoder exhibit superior performance in semantic similarity tasks in comparison to representations obtained from monolingual encoders like language modeling. The study also highlights a positive correlation between the volume of training data used for the MT-LSTM and the performance of the downstream NLP tasks.

The operational methodology involves feeding GloVe embeddings of English words to the attentional sequence-to-sequence bidirectional LSTM model. Upon training the MT-LSTM, the output from the encoder is designated as CoVe. Essentially, GloVe word vectors are employed to generate these context vectors (CoVe). In particular tasks like classification and question answering, the GloVe vector of a word is concatenated with its corresponding CoVe vector. The amalgamation of CoVe and GloVe exhibits a higher validation performance across all tested classification and question-answering tasks than models using only GloVe. Moreover, appending character n-gram embeddings further augments the performance, demonstrating that the information encapsulated by CoVe complements both the word-level insights provided by GloVe and the character-level data conveyed through character n-gram embeddings.

Universal Language Model Fine-Tuning (ULMFiT)

ULMFiT, as proposed in [66], uses Language Modeling (LM) for pretraining, establishing LM as an optimal source task due to its ability to capture myriad language facets including hierarchical relations, long-term dependencies, and semantic and syntactic aspects. Unlike Machine Translation, data for LM is copiously available which facilitates the pretraining process.

In contrast, models like ELMo [62] and CoVe necessitate training the primary/test task model from scratch while treating the pre-trained embeddings as fixed parameters, which curtails their utility. To surmount this limitation, the ULMFiT approach introduces a fine-tuning transfer technique to eschew task-specific modifications and the need for training downstream tasks from scratch—a process that demands extensive datasets and protracted convergence times.

The advancement in ULMFiT arises from its recognition that different layers encapsulate varying types of information, necessitating distinctive extents of fine-tuning. It unveils a pioneering fine-tuning approach termed discriminative fine-tuning, which permits the tuning of each layer with differentiated learning rates. This conceptually simple but potent adjustment facilitates the tailoring of parameters to task-specific features. In tandem, ULMFiT introduces Slanted Triangular Learning Rates (STLR), a scheme which initially elevates the learning rate linearly before subjecting it to a linear decay. This mechanism aids in adapting the parameters more effectively to the specific features of the task at hand.

Moreover, ULMFiT advances a gradual unfreezing technique to mitigate the risk of catastrophic forgetting that could transpire if all layers were fine-tuned simultaneously. This technique embarks on a cautious path of unfreezing, beginning with the last layer—presumed to hold the least general knowledge—and progressively unfreezing and fine-tuning all preceding layers for one epoch, in a sequenced manner until all layers are fine-tuned and convergence is achieved in the final iteration.

Collectively, the amalgamation of discriminative fine-tuning, slanted triangular learning rates, and gradual unfreezing techniques not only fortifies the ULMFiT method but also significantly enhances its performance across a spectrum of datasets. This systematic approach underscores ULMFiT's potential in efficiently navigating the intricacies of pretraining and fine-tuning in language modeling tasks.

Embeddings from Language Models (ELMo)

ELMo [62], introduces a technique to obtain word embeddings using a bidirectional LSTM model, offering a richer representation of words by encapsulating deeper and context-dependent facets of word meanings. Unlike conventional word type embeddings, ELMo's word representations are influenced by the entire input sentence, providing a more contextual understanding of each word.

The uniqueness of ELMo embeddings lies in their generation through a bidirectional LSTM, trained on an extensive text corpus employing a Language Model objective. By utilizing the internal states of a deep bidirectional Language Model (biLM), ELMo amalgamates the representations from both forward and backward language models, delivering a more nuanced word representation. These embeddings, being the output of all internal layers of the biLM, showed superior performance compared to using just the output of the top LSTM layer.

The bidirectional LSTM network in ELMo is trained with a coupled Language Model objective, and the embeddings are computed atop two-layer biLMs with character convolutions, which are a linear function of the internal network states. This process ensures that ELMo representations are deeper than traditional word vectors, as they encapsulate information from all internal layers of the bidirectional language model. Within ELMo, the higher-level LSTM states discern semantic and context-dependent aspects, while the lower-level LSTM states concentrate on syntactic features. ELMo extracts these context-sensitive features from both left-to-right and right-to-left language models, creating a comprehensive contextual representation for each token by concatenating these bidirectional representations.

Through this approach, ELMo embeddings encapsulate the intricate attributes of word usage including syntax, semantics, and the variation of usage across different linguistic contexts, effectively addressing polysemy. The paper illustrates how ELMo embeddings significantly enhance state-of-the-art performance across six demanding Natural Language Processing (NLP) chal-

lenges like question answering, textual entailment, and sentiment analysis. The comparative analysis revealed that for tasks where direct comparisons were viable, ELMo surpassed CoVe, which computes contextualized representations using a neural machine translation encoder. The superiority of ELMo in modeling complex linguistic characteristics demonstrates its potential in advancing the field of word embeddings and contributing substantially to tackling diverse NLP problems.

Generative Pre-Training (GPT)

The aspiration to amplify performance on discriminative tasks through unsupervised pre-training has been a longstanding objective within the Machine Learning research domain. (Radford et al., 2018) [63] introduced GPT, which adopted a semi-supervised methodology aimed at language understanding tasks, orchestrating a blend of unsupervised pre-training followed by supervised fine-tuning. The distinctive trait of the GPT setup is its absence of dependency on the domain congruence between target tasks and the unlabeled corpus utilized for pre-training.

The process initiates with the application of a Language Modeling objective on the unlabeled data to ascertain the embeddings, which are subsequently tailored to cater to the target tasks by employing the respective supervised objectives. GPT utilized a dataset replete with extensive stretches of contiguous text for pre-training, facilitating the generative model in learning to condition on long-range information. This approach propelled GPT to accomplish new state-of-the-art results on 9 out of the 12 datasets evaluated.

Furthermore, the paper focused into an analysis concerning the impact of various transferred layers from the pre-trained model on the accuracy of downstream tasks, notably RACE and MultiNLI. The analysis revealed a notable enhancement in accuracy with each additional transformer layer [11], signifying that each layer encapsulated essential functionality conducive to resolving target tasks.

To unravel the effectiveness of language model pre-training within transformers, the authors orchestrated a zero-shot learning experiment. The outcomes

inferred that the attentional memory within the transformer [11] considerably bolstered transfer efficacy in comparison to LSTMs.

By pre-training on a diversified corpus encompassing long stretches of contiguous text, GPT amassed significant world knowledge alongside the capability to process long-range dependencies. These acquired attributes were subsequently and successfully channelized towards resolving discriminative tasks such as question answering, semantic similarity assessment, entailment determination, and text classification, underscoring the potential of unsupervised pre-training in augmenting performance on discriminative tasks.

Bidirectional Encoder Representations from Transformers (BERT)

Pre-trained word embeddings have a recognized advantage over starting the training process from scratch. Traditionally, a left-to-right Language Modeling (LM) objective has been utilized for pre-training. In contrast, BERT advances this process by employing a bidirectional Transformer encoder, amalgamating both left and right context to enhance masked word predictions. (Devlin et al., 2018) [64] focus into pre-training deep bidirectional Transformer encoder representations using unlabeled text, jointly conditioned on both left and right contexts across all layers. The process within the BERT framework is bifurcated into pre-training and fine-tuning stages.

Initially, during pre-training, the model is educated on unlabeled data across diverse tasks. The subsequent fine-tuning stage initiates with the BERT model pre-loaded with pre-trained parameters, which are then fine-tuned using labeled data derived from downstream tasks. Each of these tasks possesses uniquely fine-tuned models, albeit initialized with identical pre-trained parameters.

A noteworthy assertion by the authors is the potential detrimental effect of a unidirectional approach on sentence-level and token-level downstream tasks. This is exemplified in question-answering tasks where assimilating context from both directions is pivotal. Unlike GPT, which is confined to a left-to-right architecture, BERT transcends this limitation by considering context from both directions. It achieves this through the Masked Language Model

(MLM) pre-training technique, where certain tokens are randomly masked and the model is tasked with predicting these masked tokens based on their context. This strategy facilitates the fusion of both left and right context within the representation. Moreover, BERT incorporates a Next Sentence Prediction (NSP) task in its pre-training regimen.

BERT's MLM methodology fosters deep bidirectional representations, distinguishing it from ELMo which employs a shallow concatenation of independently learned embeddings. Given that crucial downstream tasks like Question Answering (QA) and Natural Language Inference (NLI) necessitate understanding the interrelationship between sentences, BERT's binarized next-sentence prediction task in pre-training is instrumental. The results showcased in Table 8 elucidate how BERT surpassed GPT and ELMo across all GLUE tasks, with BERT-large exhibiting superior performance over BERT-base. These findings accentuates the significant merit of bidirectional pre-training for obtaining more enriched representations, laying a solid foundation for enhanced performance in various downstream tasks.

Unified Pre-trained Language Model (UNILM)

While BERT excels in boosting performance for a vast array of natural language understanding (NLU) tasks, its bidirectional nature limits its application in natural language generation (NLG) tasks. To bridge this gap, (Dong et al., 2019) [67] introduce UNILM, a multi-layer Transformer network designed for both NLU and NLG tasks. UNILM is jointly pre-trained on substantial text data, aiming at three unsupervised language modeling objectives while sharing a common set of parameters. This pre-trained unified model is then fine-tuned and evaluated on diverse datasets encompassing both understanding and generation tasks. Similar to BERT, UNILM can be fine-tuned, and if necessary, augmented with task-specific layers to cater to various downstream tasks.

Contrary to BERT's primary focus on NLU tasks, UNILM's flexibility allows it to consolidate context for different language model types, rendering it suitable for both NLU and NLG tasks. A key advantage of UNILM is

its parameter sharing feature, which results in more generalized text representations as they are concurrently optimized for varied language modeling objectives, thereby reducing overfitting risks.

Moreover, UNILM's application extends beyond NLU tasks; its sequence-to-sequence language modeling capability makes it a favorable choice for NLG tasks. It employs a masking technique to manage the amount of context a token attends to while computing its contextualized representation. Upon pre-training, these representations are further fine-tuned using task-specific data from downstream tasks. The vector representation of each input token is formulated by aggregating the corresponding token, position, and segment embeddings. By utilizing bidirectional LM pre-training, UNILM is able to encapsulate contextual information from both directions, thereby generating enhanced contextual text representations compared to unidirectional approaches. In summary, UNILM presents a versatile solution, expanding the scope of both natural language understanding and generation tasks beyond the capabilities offered by BERT.

2.2.10 Fine-Tuning-based Embeddings

Typically, word embedding research is centered on general-domain text generation. However, (Chiu et al., 2016) [68] highlight that this approach falls short in domain-specific analysis of large datasets, such as in the legal, financial sectors. In the following, we present some different word embedding variations within the NLP realm, focusing on fine-tuned embeddings that adapt and adjust to contextual shifts and downstream tasks.

Contrary to past methods where embeddings remained static post-input, the new breed of embeddings evolves when applied to downstream tasks. Pre-trained Language Models (LMs) use extensive text data to learn contextualized text representations by predicting words based on their surrounding context. These models can then be fine-tuned to align with downstream tasks. Once transferred, the embeddings undergo fine-tuning on the target

task data to absorb task-specific features, thereby enhancing performance on downstream tasks.

While pre-training usually employs generic-domain data and tasks, the data concerning target tasks often come from disparate distributions. This necessitates the fine-tuning of the pre-trained LM on target task data to acclimate to the unique aspects of said data. Notably, this fine-tuning approach reaches convergence faster than initiating training from scratch. In such a system, embeddings birthed from pre-training serve as input to the model, which then undergoes refinement to boost performance on downstream tasks.

The categorization of fine-tuned embeddings based on downstream tasks can be nuanced. Cross-Lingual Embeddings are designed to bridge language barriers by facilitating mapping across languages. On the other hand, Knowledge-Enriched Embeddings meld external knowledge to better semantic understanding. Domain-Specific Embeddings are tailored to resonate with the lexicon and semantics of particular fields. Multi-Modal Embeddings amalgamate information from diverse modalities like text and images for more robust representations. Lastly, Language-Specific Embeddings are honed to the linguistic peculiarities of specific languages, ensuring a custom-fit for varying linguistic landscapes.

In conclusion, fine-tuning word embeddings to the context and tasks at hand, especially in domain-specific scenarios, paves the way for more accurate and insightful analysis, taking NLP applications a notch higher in performance and relevance.

Cross-Lingual Embeddings

The research predominantly focuses on pre-training models using large unlabeled corpora in English, followed by fine-tuning them on specific tasks with smaller English supervised datasets, representing a monolingual approach. However, a shift towards a cross-lingual method has emerged, wherein the pre-training and fine-tuning are conducted in different languages, resulting in multi-language spanned embeddings. Such encoders, when trained multilingually, can encode sentences from multiple languages into a shared semantic

space. These cross-lingual models can be categorized into cross-lingual understanding (XLU) and cross-lingual generation (XLG) models, with several prevalent models under each category.

For instance, Google's Multilingual NMT (M-NMT) [69] employs a standard NMT system which, owing to shared parameters across language pairs, generalizes well beyond language boundaries, significantly improving the quality of low-resource language pairs. Following pre-training on 12 language pairs, this model exhibited capabilities for zero-shot translation and genuine transfer learning, translating between language pairs unseen during training.

Contrary to BERT, which is trained only on English corpus, M-BERT is trained on Wikipedia pages from 104 languages using a shared word-piece vocabulary. Despite lacking a cross-lingual objective or parallel data, M-BERT [70] surprisingly performed well in cross-lingual generalization, enabling task-specific annotation transfer from source to target languages, even across different scripts.

Adding to this, the XML technique [71] introduces a Translation Language Modeling (TLM) objective to enhance cross-lingual pre-training. Unlike traditional methods, TLM concatenates parallel sentences from source and target languages, masking random words in both to foster alignment between representations of both languages.

Similarly, Unicoder [72], proposed in another study, employs three cross-lingual tasks for pre-training, utilizing a Machine Translation dataset to train various tasks like cross-lingual word recovery and cross-lingual paraphrase classification. This approach aided Unicoder in becoming a better language-independent encoder and learning language mappings from multiple angles.

On the other hand, XLG, unlike NMT, entails summary or response generation in the target language based on source language documents. Models like XLM-RoBERTa (XLM-R) [73] and MASS [74] have shown significant gains in cross-lingual benchmarks and various language generation tasks respectively.

XNLG, as discussed in [75], employs monolingual data for masked language modeling (MLM) and a cross-lingual setting for cross-lingual MLM (cross-MLM) during pre-training. This method uses bilingual sentence alignment to encode cross-lingual texts into a shared embedding space. Post pre-training, the model is fine-tuned on downstream natural language generation (NLG) tasks with monolingual data. This approach outperforms machine translation-based methods in cross-lingual question generation and abstractive summarization tasks, significantly enhancing performance for low-resource languages by leveraging rich-resource language information.

These varied approaches and models elucidate the evolving landscape of cross-lingual embeddings, catering not only to language understanding but also to language generation across a multitude of languages. This evolution is gradually bridging the gap between languages, fostering a more inclusive and accessible multilingual NLP domain.

Knowledge-Enriched Embeddings

Contextual word representations, devoid of any real-world entity information, lack the capability to recall or retrieve factual knowledge about those entities. This necessitates the infusion of knowledge information during the pre-training tasks of most language models, enriching them with structured factual data, and subsequently enhancing language understanding. Knowledge-Enriched Embeddings (KEE) emerge as a solution, offering word embeddings supplemented with information from external sources like ontologies or knowledge graphs, making them potent for knowledge-driven applications such as entity typing and relation classification and extraction. This knowledge information is structurally captured using a knowledge graph in a triplet (h, r, t) format, which outlines a relational fact with a head entity (h), a tail entity (t), and the relation type (r). Through the generation of Knowledge Embeddings (KE) using knowledge graphs (KGs), entities along with their relationship information are effectively embedded.

One notable approach is ERNIE [76], which takes the pre-training of a language representation model a notch higher by utilizing both large-scale tex-

tual corpora and KGs, maximizing the utilization of lexical, syntactic, and knowledge information. By proposing a new pre-training task coined as denoising entity auto-encoder (dEA), ERNIE introduces knowledge into language representation through informative entities. In this mechanism, certain token-entity alignments are randomly masked, prompting the system to predict corresponding entities based on aligned tokens. ERNIE comprises not just a Textual Encoder (T-Encoder) that captures lexical and syntactic nuances, but also a Knowledgeable Encoder (K-Encoder) that aggregates token and entity embeddings. Through this approach, ERNIE significantly augments the performance on knowledge-driven tasks like Entity Typing and Relation Classification.

On the other hand, KnowBert [62] proposes a method to embed multiple knowledge bases (KBs) into large-scale models, enhancing their representations with structured, human-curated knowledge. KnowBert's Knowledge Attention and Recontextualization (KAR) method first identifies entity spans in raw text, retrieves their embeddings from KB to construct knowledge-enhanced entity span representation, and recontextualizes these representations with word-to-entity span attention, facilitating long-range interaction between entity spans in context. This enhancement showcases improved perplexity and recall of entity facts, with downstream evaluations indicating better performance over relationship extraction, entity typing, and word sense disambiguation tasks.

A distinctive approach SentiLARE [77], aimed at bolstering language understanding and aiding downstream tasks in sentiment analysis. By introducing a new label-aware MLM pre-training task with two subtasks for early fusion and late supervision, it effectively integrates word-level linguistic knowledge such as PoS tags and sentiment polarity into pre-trained models. The devised sub-tasks forge a connection between sentence-level representation and word-level linguistic knowledge, proving beneficial for sentiment analysis tasks as corroborated by experiment outcomes showing superior performance across various sentiment analysis tasks.

K-BERT [78] addresses the concern of Knowledge Noise, which may arise from an excessive incorporation of knowledge, potentially altering the intended meaning of sentences. By introducing a soft position and visible matrix, K-BERT manages to limit the impact of knowledge, transforming the original sentence into a knowledge-rich sentence tree while maintaining the original meaning. This approach demonstrates significant outperformance over BERT, especially in domain-specific tasks, showcasing its potential in tackling knowledge-driven problems requiring expert inputs.

Contrarily, KEPLER (KE) [79] adopts a unique strategy by utilizing entity descriptions to encode and map text and entities to a unified semantic space, without the necessity of a separate KE model. This approach not only eliminates the need for an entity linker but also circumvents the additional inference overhead. KEPLER, by encoding entities from their descriptions, seamlessly bridges the gap between KE and PLM, aligning the semantic space of text with the symbol space of entities in KGs. The model showcases its ability to generate embeddings for unseen entities based on their descriptions, a feat unachievable by conventional KE methods. The integration of factual knowledge from external KGs into language representations through joint training of KE and MLM objectives remarkably improves NLP and KE applications, like relation extraction and entity typing.

Further notable contributions include the knowledge-enriched word embeddings (KEWE) provided by the authors in [80], which use knowledge graphs to encode reading difficulty knowledge into words, evaluated on both English and Chinese datasets. In the biomedical domain, the authors in [81] provide empirical evidence suggesting the improvement in word embeddings quality when an external semantic knowledge base is combined with local contextual information for generating biomedical concepts. Additionally, a Knowledge-enriched answer generator (KEAG) is proposed by the authors in [82], which exploits an external symbolic knowledge base to generate answers. This variety of KEE models demonstrates the potential of integrating external knowledge sources with local contextual information to significantly enhance the quality of word representations, thereby substantially benefiting

a plethora of NLP tasks such as entity recognition, relation extraction, and semantic labeling.

Domain-Specific Embeddings

Embeddings generated from large generic corpora like Wikipedia may not perform well in specialized domain tasks due to the lack of domain-centric word distribution, especially observed in fields like biomedicine. Recent studies have shifted towards utilizing domain-specific corpora for pre-training to achieve more tailored embeddings in various domains such as finance, healthcare, and reviews [83], sentiments [84], emotions [85]. Notable research [86, 87] demonstrated that domain-tailored embeddings, generated from relevant sectors like biomedicine, oil/gas, and social media (Covid-19 Tweets), enhanced performance in corresponding NLP tasks. A study [88] showed significant improvement in multi-label classification tasks when a shallow neural network was trained on a large Radiology-related corpus.

BioBERT [89], SciBERT [90], ClinicalBERT [91] and PatentBERT [92] are prime examples of models pre-trained on domain-specific corpora that outperformed the original BERT in biomedical, scientific, clinical, and patent classification NLP tasks respectively. SentiBERT [93], designed to capture compositional sentiment semantics, effectively merged contextualized representation with syntactic tree structure, showcasing a better capture of semantic compositionality. The shift towards domain-specific pre-training indicates a significant stride towards achieving improved domain-centric embeddings and performance in respective NLP tasks.

Multimodal Embeddings

The advancement of multimodal embedding models showcases the ability to amalgamate information from audio, visual, and text modalities to foster enhanced word embeddings through deep neural networks. Large neural networks like [11, 64, 94] have marked significant progress in multimodal settings.

In the realm of speech-based embeddings, generating embeddings for audio words presents a challenge due to varying audio signal realizations of the

same word token in different utterances. A notable attempt to overcome these challenges is SpeechBERT [95], which tackles the end-to-end Spoken Question Answering (SQA) task by learning audio and text jointly. It employs Text-BERT to extract semantic information from text data and utilizes an RNN sequence-to-sequence autoencoder to capture the phonetic structure of audio words. SpeechBERT managed to achieve state-of-the-art results on the Spoken SQuAD dataset.

Turning towards image-based embeddings, the objective is to align text elements with regions in images. VisualBERT [96] and ViLBERT [97] employ joint processing of image and text inputs and separate streams for language and vision processing respectively to enrich the interaction between words and objects. B2T2 [98], LXMERT [99], and VL-BERT [100] also offer intriguing approaches in bridging visual and textual modalities, with strategies like multimodality-based masking and single-stream input processing. Unicoder-VL [101] use a cross-modal pre-training framework to learn joint representations of visual and linguistic contents.

On the video-based embedding front, VideoBERT in [102] represents videos using visual words or tokens, focusing on learning and forecasting videos along with aligning text and video domains. Contrastingly, CBT in [103] opts for training each modality separately and maximizes mutual information between modalities at the sequence level. VideoTranslate (VideoAsMT) [104] conceptualizes video understanding as a machine translation task, employing an encoder-decoder architecture to generate text from multimodal representations. UniVL [105] and HERO [106] extend the landscape with their flexible model structures and hierarchical encoding of multimodal inputs, each targeting a nuanced understanding and generation of text and video modalities.

In essence, these multimodal embedding models, with their varied techniques, are stepping stones towards a robust integration of audio, visual, and textual information which is fundamental for enhancing the capabilities of deep neural networks in understanding and generating multimodal data.

Language-Specific Embeddings

Domain-specific embeddings are specialized word embeddings trained on large corpora within particular domains to seize the relationships between terms and expressions pertinent to that domain. These domain-centric models can significantly boost the accuracy and performance of NLP tasks like text classification, sentiment analysis, and named entity recognition within those domains. Typically, the pretraining phase of attention-based models unfolds in an unsupervised manner on vast domain-specific corpora, which is then followed by fine-tuning on a smaller labeled dataset pertinent to the domain.

Compared to general-purpose models, recent studies have demonstrated that domain-specific models yield superior performance on NLU tasks within their respective domains. These models [76, 107–119], largely built on the BERT architecture, undertake the same pre-training tasks (MLM and NSP). Moreover, most of these models have formulated their own pre-training and test datasets, along with domain-specific benchmarking frameworks. Such domain-tailored models contend that pretraining on smaller domain-specific datasets can achieve comparable or even superior results relative to training on larger general-purpose datasets. For evaluating the performance of domain-centric NLP models, benchmarks designed for those specific domains are utilized. Analogous to how GLUE and SuperGLUE benchmark frameworks are used for general English NLP tasks, domain-specific benchmarks are employed to assess models tailored for domains like medical, legal, or financial text analysis.

2.3 Traditional Machine Learning Approaches

Websites and Web pages classification has been extensively studied, with researchers exploring different approaches and techniques to improve classification performance. In the following we group the existing works following traditional machine learning based techniques.

2.3.1 Naive Bayes (NB)

The Naive Bayes classifier is a popular probabilistic model used for text classification due to its simplicity and effectiveness. It operates on the principle of conditional independence between features, making it suitable for large datasets. This section explores various studies that have applied and enhanced the Naive Bayes algorithm for web page classification, demonstrating its versatility and performance improvements through different modifications and feature selection techniques.

(Li et al., 2005) [120] introduce a method for web page categorization using a hybrid neural network architecture. They represent web pages as feature vectors, weighted by term frequency and sentence significance, with PCA for feature selection and SOFM for classification, showing substantial improvements over KNN and Naive Bayes. (Petprasit et al., 2015) [121] propose a methodology for classifying E-commerce web pages using MLP neural networks, achieving 97.60% accuracy, outperforming Naive Bayes, RBF, and SVM. (Zhang et al., 2018) [122] develop a neural network model for detecting phishing websites using deep learning and transfer learning, achieving 86% accuracy, surpassing SVM and decision tree methods.

(Tomar et al., 2006) [123] present a modified Naive Bayesian method for web page categorization, using TF-IDF for word relevance, reducing computation time, and improving accuracy. (Soon et al., 2010) [124] evaluate Naive Bayes, Balanced Mixed Classification, and Mixed Classification with Priority to the Statistical Method for automated information extraction from newspaper advertisements, highlighting Naive Bayes efficiency. (Shaohong et al., 2011) [125] explore web page classification using HTML tags and terms, showing Naive Bayes outperforming SVM and KNN.

(Haleem et al., 2014) [126] employ the SPaC-MR classifier for text classification, evaluated alongside SVM, J48, Naive Bayes, and PART, and propose the improved SPaC-NF, demonstrating superior performance. (Zhang et al., 2011) [127] address phishing attacks using texture-based features and classi-

fiers, including SVM, Naive Bayes, and neural networks, achieving promising results. (Hu et al., 2007) [128] combine text-based and image-based features with classifiers like SVM, Decision Trees, and Bayesian Networks for recognizing pornographic web pages, achieving accurate identification.

(Fersini et al., 2008) [129] enhance web document classification by evaluating image-blocks and textual terms with classifiers such as Naive Bayes, SVM, Decision Trees, and KNN, showing improved accuracy over traditional TFxIDF models. (Suganya et al., 2017) [130] use the Cross Training based Corrective (CTC) approach, enhancing classifier performance through cooperative adjustment of web page types, demonstrating potential improvements with KNN, SVM, and Naive Bayes classifiers. (Abbas et al., 2016) [131] investigate a rule-based methodology for web page classification, utilizing word frequency, association rule mining, and category-specific keywords, providing insights into effective classification with potential Naive Bayes application. (Patil et al., 2012) [132] review features and classifiers for web page classification, including SVM, neural network agents, and decision rules, highlighting efforts to optimize classification performance using diverse features and classifiers.

2.3.2 K-Nearest Neighbors (KNN)

The KNN algorithm is a non-parametric method widely used for classification tasks. It classifies data points based on the majority class among its k-nearest neighbors. This section reviews advancements in the KNN approach, including enhanced feature selection, the incorporation of implicit links, and various post-classification correction methods, showcasing its application in web page classification and related tasks.

(Kwon et al., 2000) [133] introduce LIC, an enhanced KNN approach that incorporates feature selection, term-weighting using markup tags, and a revised document-document similarity measure. Experimental results show significant performance improvements over traditional KNN. (Xu et al., 2011) [134]

propose a method using link information to classify web pages, exploiting parent page reference information instead of relying solely on web content, enhancing classification accuracy.

(Abdelbadie et al., 2014) [135] enhance KNN and other text classifiers through Clique Based Correction (CBC), which uses implicit links between web pages to form cliques for category rectification, improving accuracy. (Zheng et al., 2015) [136] introduce the LWCS system, combining anchor graph hashing with kNN for large-scale web page classification, demonstrating fast and storage-efficient performance.

(Belmouhcine et al., 2016) [137] use implicit links from query logs combined with KNN to compute similarities based on click frequencies, enhancing effectiveness through a two-level ranking process. (Kwon et al., 2003) [138] employ various features and classifiers, such as bag-of-words, N-grams, TF-IDF, and link-based features, with Naive Bayes, SVM, memory-based reasoning, decision trees, and neural networks for accurate classification.

(Kameshwari et al., 2017) [139] evaluate a different set of classifiers including KNN, SVM, and Naive Bayes, likely considering features such as Bag-of-Words, TF-IDF, N-grams, and word embeddings to enhance classification results. (Zheng et al., 2015) [136] conduct experiments with the LWCS system for Chinese web pages, analyzing the parameter "k" in KNN and comparing hash-based methods with the original vector-based method, demonstrating fast and accurate performance.

(Li et al., 2005) [120] use Principal Component Analysis (PCA) and Self-Organizing Feature Map (SOFM) for feature extraction in web sports news classification, achieving higher precision, recall, and F1 measures compared to Naive Bayes and KNN. (Jiao et al., 2010) [140] incorporate the hyperlink factor into their web page classification approach, significantly enhancing accuracy by using mutual information to measure feature relevance.

2.3.3 Decision Tree (DT)

Decision Tree algorithms are intuitive and powerful tools for classification that split data into subsets based on feature values, forming a tree-like structure. This section presents various implementations of Decision Tree classifiers for web page classification, highlighting improvements in feature discretization, ensemble methods, and adaptations for handling structured attributes, which enhance classification accuracy and efficiency.

(Mangai et al., 2012) [141] implement an algorithm that converts features into discrete values for web page classification, significantly improving accuracy compared to using continuous features. (Jelodar et al., 2015) [142] assessed various decision tree-based algorithms for distinguishing between non-advertisement and advertisement websites, finding that the J48 algorithm outperformed others such as Decision Stump, Hoeffding tree, Logistic model tree (LMT), Random Forest, Random tree, and REP Tree in terms of accuracy and effectiveness.

(Andoohgin et al., 2018) [143] propose a technique for identifying hijacked journals using a classification algorithm trained on a dataset of hijacked and authentic journals, utilizing nine specific features to construct a decision tree for effective detection.

(Dadkhah et al., 2015, 2016) [144] and [145] introduce features for detecting hijacked journals, such as domain rank, domain age, and consistency between the server's country and the journal's country. They employed various classification algorithms including Decision Stumps, J48, Random Tree, and REP Tree using the WEKA software, with the Random Tree algorithm achieving the lowest error rate.

(Estruch et al., 2006) [146] present the distance-based decision tree (DBDT), which handles structured attributes like lists, graphs, and sets, using a splitting criterion based on the metric condition "is nearer than." This allows for more flexible and nuanced decision-making within the decision tree learning process.

2.3.4 Support Vector Machines (SVM)

Support Vector Machines (SVM) are robust classifiers known for their ability to handle high-dimensional data and create optimal hyperplanes for classification. This section covers a range of SVM-based approaches for web page classification, including dual feature space integration, content-based and visual feature combinations, and advanced kernel methods, illustrating the versatility and effectiveness of SVMs in diverse classification scenarios.

(Alvari et al., 2017) [147] developed a semi-supervised learning model using a dual feature space approach, combining six feature groups from ads and geometric properties captured through a data adjacency graph. This advanced SVM classifier integrated both feature spaces and outperformed existing solutions. (Tian et al., 2013) [148] introduced a method for classifying images on websites by integrating text and visual features, employing SVM classifiers for each feature type and a multiple-classifier fusion technique, demonstrating superior performance compared to single classifiers.

(Abbasi and Chen, 2009) [149] proposed AZProtect, using a stack of SVM classifiers to detect and classify fake websites based on content similarity and duplication features. This solution showed promising results when trained on 500 websites. (Fersini et al., 2008) [150] enhanced term selection and weighting by identifying important image-blocks within web pages, with SVM outperforming other classifiers in precision and recall metrics.

(Ulges et al., 2011) [151] employed color-enhanced visual word features and SVM classifiers to detect child pornography, improving automatic detection through histograms of visual words. (Rowley et al., 2006) [152] addressed large-scale image-based adult-content filtering by integrating various features into an SVM classifier with RBF kernels, achieving a 50% detection rate for adult-content images with a 10% false positive rate.

(Sun et al., 2003) [153] improved web classification by incorporating text and context features like hyperlinks and HTML tags, finding that SVM classifiers with title and anchor words significantly improved accuracy. (Kan et

al., 2004) [154] proposed a two-phase pipeline using SVM classifiers, demonstrating that URL-only features significantly enhanced performance.

(Belmouhcine et al., 2015a) [155] introduced an implicit links-based Gaussian kernel for SVM classification, improving performance by incorporating user's intuitive judgments. In another study, (Belmouhcine et al., 2015b) [156] combined web page representations from implicit graph structures with various classifiers, significantly improving accuracy.

(Gu et al., 2016) [157] used Latent Dirichlet Allocation (LDA) to extract features for web page classification, enhancing effectiveness when combined with SVM. (Chen et al., 2006) [158] employed Latent Semantic Analysis (LSA) and Web Page Feature Selection (WPFS) with SVM classifiers, finding the anova kernel function most effective. (Soon et al., 2010) [124] explored Information Extraction (IE) patterns for web classification using the C5.0 algorithm, achieving significant accuracy and suggesting further research into Association Rule Mining (ARM).

Table 2.6: Summary of Traditional ML based Approaches

Approach	WS/WP	Dataset	TR	Classifier	Results
Andoohgin et al. (2018) [143]	Website	Dataset of 104 journal websites	Various features: number of broken links, number of published articles in a year, consistency between server and journal country, existence of inactive links, use of "-" in URL, rank in search engines, age of domain, countries visiting the website, journal aim and scope	Decision Stump, J48, Random Tree, Random Forest	Decision Stump: Error rate: 17.31%, J48: Error rate: 8.65%, Random Tree: Error rate: 0%, Random Forest: Error rate: 0.96%

Approach	WS/WP	Dataset	TR	Classifier	Results
Jelodar et al. (2017) [159]	Website	Custom dataset	TF-IDF, n-grams	SVM, Naive Bayes, Decision Tree	SVM: Accuracy: 92%, Naive Bayes: Accuracy: 88%, Decision Tree: Accuracy: 85%
Alvari et al. (2017) [147]	Website	Custom dataset (web pages)	Content features, structure features	Semi-supervised learning, SVM	Accuracy: 85%, F1: 0.84
Kameshwari et al. (2017) [139]	Web Page	Open Directory Project (ODP)	Text features	SVM, Naive Bayes, KNN	SVM: Accuracy: 87.9%, Naive Bayes: Accuracy: 89.0%, KNN: Accuracy: 85.1%
Gu et al. (2016) [157]	Web Page	Custom dataset	Content features, link features	SVM, Decision Tree	SVM: Accuracy: 89%, F1: 0.87, Decision Tree: Accuracy: 85%, F1: 0.83
Belmouhcine et al. (2016) [137]	Web Page	Custom dataset	Implicit links-based kernel, content features	SVM	Accuracy: 91%, Precision: 0.90, Recall: 0.89

Approach	WS/WP	Dataset	TR	Classifier	Results
Jelodar et al. (2015) [142]	Website	Custom dataset prepared using search engine results	Text features	J48, Decision Stump, HoeffdingTree, LMT, RandomForest, RandomTree, REPTree	J48: Accuracy: 71.43%, Precision: 0.682, F-measure: 0.679
Zheng et al. (2015) [136]	Web Page	Ifeng.com	TF-IDF	KNN	Improved performance with KNN
Belmouhcine et al. (2015) [156]	Web Page	Custom dataset	Implicit links-based kernel, content features	SVM	Accuracy: 90%, Precision: 0.89, Recall: 0.88
Wang et al. (2013) [160]	Website	WEBSPAM-UK2006	Content features (72), Link features (81)	LC-training, Link-training2, Co-training, AdaBoost, Decision Tree	LC-training: F1: 0.807, AUC: 0.941, Link-training2: F1: 0.816, AUC: 0.948
Tian et al. (2013) [148]	Website	Google Image Search	VSM, GLCM	SVM, Bayesian Network	Accuracy: 85%, F1: 0.84

Approach	WS/WP	Dataset	TR	Classifier	Results
Jayanthi et al. (2013) [161]	Website	Custom dataset (emails)	TF-IDF, POS tagging	SVM, Naive Bayes, Decision Tree	SVM: Accuracy: 91%, F1: 0.89, Naive Bayes: Accuracy: 87%, F1: 0.85, Decision Tree: Accuracy: 83%, F1: 0.81
Mangai et al. (2012) [141]	Web Page	WebKB	Discretization	OneR, ID3, J48, Naive Bayes, Kstar	Naive Bayes: Accuracy: 97.09%, Kstar: Accuracy: 96.43%
Egele et al. (2011) [162]	Website	PhishTank, OpenPhish	URL features, host-based features	SVM, Random Forest	SVM: Accuracy: 95%, F1: 0.94, Random Forest: Accuracy: 93%, F1: 0.92
Ulges et al. (2011) [151]	Website	Custom datasets (pornography, CSA)	Image and text features	SVM	Precision: 92%, Recall: 90%, F1: 0.91
Xu et al. (2011) [134]	Web Page	Sohu, Netease, Yahoo, Javaeye, CSDN	Vector Space Model, link information	LIC, KNN, SVM	LIC: Accuracy: 99.6%, SVM: Accuracy: 85.2%

Approach	WS/WP	Dataset	TR	Classifier	Results
Soon et al. (2010) [124]	Web Page	WebKB	Text features	SVM, Decision Trees	Accuracy: 78%, F1: 0.75
Jiao et al. (2010) [140]	Web Page	Custom dataset	Text features	Not specified	Improved classification accuracy by 10%
Fersini et al. (2008) [150]	Web Page	10,000 web pages from Yahoo! Directories	Term frequency-weighting schemes, image-block analysis	SVM, Multinomial Naive Bayes	SVM: F1: 0.88, Naive Bayes: F1: 0.82
Abbasi & Chen (2009) [149]	Website	Fake websites from online trading community databases	Fraud cues (URLs, keywords)	SVM	Accuracy: 91%, Recall: 92%, Precision: 89%
Chen et al. (2006) [158]	Web Page	Sports news dataset	Latent Semantic Analysis (LSA), Web Page Feature Selection (WPFS)	SVM	Accuracy: 82.5%
Rowley et al. (2006) [152]	Web Page	Custom dataset	Feature extraction (text, image, links)	SVM, Decision Trees, Neural Networks	Error rates: SVM best with 4.6%

Approach	WS/WP	Dataset	TR	Classifier	Results
Estruch et al. (2006) [146]	Web Page	Custom dataset (sports news)	Graph-based representation, content-based metrics	DBDT algorithm	F1 measure: 0.764
Qi et al. (2006) [163]	Web Page	ODP dataset, WebBase dataset	TF-IDF, Context features (title, URL)	SVM, Bayesian network model	ODP: Accuracy: 91.4%, WebBase: Accuracy: 56%
Li et al. (2005) [120]	Web Page	Custom dataset of sports news (Yahoo and Google)	Term frequency-weighting scheme, PCA for feature selection, block importance method	SOFM neural network	Precision: 86.24%, Recall: 88.12%, F1: 86.87%
Kan et al. (2004) [154]	Web Page	WebKB subset (ILP 98 dataset)	URL fragments, anchor text, title text, page text	SVM, FOIL-PILFS	F1 measure (macro-averaged): Improved across various configurations

Approach	WS/WP	Dataset	TR	Classifier	Results
Sun et al. (2003) [153]	Web Page	WebKB	Text features, title, and anchor words	SVM	Course: F1: 0.526, Faculty: F1: 0.550, Project: F1: 0.277, Student: F1: 0.655
Kwon et al. (2003) [138]	Website	Korean commercial web directory	HTML-based features (title, meta tags, anchor text)	SVM, KNN, Decision Trees	Accuracy: 80%
Kwon et al. (2000) [133]	Web Page	Korean commercial web directory	Term weighting scheme, features using markup tags	KNN	Accuracy: 84%

Table 2.6 provides a comprehensive summary of various traditional machine learning-based approaches for website and web page classification, sorted by year. The approaches utilize a range of classifiers, including DT, KNN, and SVM. Notably, SVM and NB classifiers are the most dominant, reflecting their widespread application and effectiveness in this domain. The table highlights each study's methodology, model, features, and key results, showcasing significant advancements in classification accuracy and performance through innovative feature selection, combination techniques, and integration of various data types. In the following section, we review the DL based approaches.

2.4 Deep Learning Approaches

Deep Neural Networks (DNNs) are advanced artificial neural networks that mimic the human brain's functioning. They outperform traditional models in tasks such as speech recognition, image processing, and text comprehension. The type of dataset, whether single-label or multi-label, determines how it's processed before being input into the DNN. DNNs consist of multiple interconnected layers where each layer is only connected to its immediate preceding and succeeding layers [12]. Input to these networks can be constructed using various methods like TF-IDF or word embedding. Depending on the task, the output layer may vary in size. The design of DNNs involves a back-propagation algorithm with activation functions like sigmoid or ReLU [164]. In cases of multi-class classification, a Softmax function is used for the output layer. For text classification, the raw text data is vectorized to serve as input. The effectiveness of a trained model can be assessed using tasks like sentiment analysis or question answering. Over the years, various deep learning models, such as multilayer perceptron [165] and recursive neural network [166], have been developed for text classification, showing improved performance over traditional models. Subsequent models utilized CNNs, RNNs, and attention mechanisms [167–169]. The introduction of BERT [64], which

offers contextualized word vectors, marked a pivotal advancement in NLP and text classification. Many studies [170, 171] have since built upon BERT, achieving superior performance in multiple NLP tasks. Furthermore, some researchers have explored Graph Neural Network (GNN) [172, 173] for text classification, capturing unique structural text information not achievable by other methods.

2.4.1 MultiLayer Perceptron-based Approaches

MultiLayer Perceptrons (MLPs), also known as "vanilla" neural networks, play a crucial role due to their simple yet effective architecture comprising an input layer, a hidden layer with activation functions, and an output layer, interconnected by weights denoted as w_i. Their ability to treat text as a bag of words allows them to achieve remarkable performance on various benchmarks, often outperforming traditional models. Notable among MLP-based methods. Simpler MLP architectures like deep averaging networks (DAN) [174] demonstrate comparable or superior performance to more complex methods [175] by treating input text as an unordered bag-of-words and using feature extraction techniques like TF-IDF or word embeddings. Doc2Vec [6] takes this a step further by including word order and contextual paragraph information, comparable to CBOW [59], leading to enhanced performance. These developments highlight MLP's versatility in bridging shallow and deep learning methods, laying the groundwork for early word embedding techniques while excelling as standalone classifiers.

2.4.2 CNN-based Approaches

Convolutional Neural Networks (CNNs), initially designed for image processing, have been effectively adapted for various text classification tasks, including hierarchical document classification and web page categorization [176–179]. CNNs process input data, whether images or text, using kernels to create feature maps. These maps can be layered for complex feature de-

tection, and to manage computational complexity, pooling techniques like max pooling are employed [180]. For text classification, text is converted into vector format, often using word embeddings, and then processed by convolutional filters. TextCNN [9] is a prominent example, known for its simplicity and effectiveness in text classification tasks, using a single convolutional layer and max pooling to identify discriminative phrases.

While CNNs are generally associated with speed and efficient latent representations, their application to text differs from images, as properties like location invariance are less relevant [181, 182]. In text classification, CNNs apply multiple filters in their convolutional layer, followed by pooling to produce a final text vector representation for classification [183].

Additionally, interest lies in utilizing unlabeled data during training, such as the two-view semi-supervised learning model [184]. The Deep Pyramid Convolutional Neural Network (DPCNN) [185] enhances performance by increasing network depth. Text representations in CNNs vary from character to sentence level, each catering to specific aspects of language processing, as demonstrated in models like TransCap [180].

(Shao et al.,2018) [186] and (Barcaroli et al., 2019) [187] demonstrate CNN's effectiveness in real-time web content categorization and e-commerce website category prediction, surpassing traditional methods in text feature extraction and recommending the integration of textual and image-based data for enhanced performance.

In addressing the challenges of categorizing web pages based on visual content, (Lopez et al., 2019, 2018) [188, 189] propose frameworks employing Deep Convolutional Neural Networks (DCNNs). These frameworks utilize transfer learning to overcome issues related to high computational costs and extensive training dataset requirements, proving to be efficient in constructing accurate classifiers for complex visual tasks with limited data. These methods show notable adaptability by identifying new web page categories during testing, thus eliminating the need for a complete set of web categories at the training phase. Also using DCNNs, (Nandanwar et al., 2020)

[190] present a framework for categorizing web pages based on visual content using deep learning. The proposed method utilizes the VGG-19 deep convolutional neural network (DCNN) to extract feature vectors from images on web pages. Transfer learning is applied to reduce computational costs and improve the efficiency of the model. The framework is tested against traditional handcrafted image descriptor methods, Fisher Vector (FV) and Vector of Aggregated Local Descriptor (VALD), and achieves a classification accuracy of 86%. The study demonstrates the effectiveness of using visual features for web page categorization, particularly for modern web pages that contain significant multimedia content. The authors highlight the potential benefits of this approach for applications such as cross-language information retrieval and recommend further development of hybrid models that combine visual and textual features for comprehensive web page categorization.

Addressing the critical issue of online adult content, [191–193] propose CNN-based methods for detecting pornographic images. These approaches involve advanced training algorithms, data augmentation techniques, and fast image scanning methods, all contributing to effective detection and recognition of adult content. The effectiveness of these methods is demonstrated through experiments, showing state-of-the-art performance in pornographic image detection and adult image recognition. These studies underscore the importance of deep learning methods in categorizing and analyzing web content, especially in the context of creating a safer online environment.

(Alsaade and Alzahrani, 2022) [194] propose a CNN and transfer learning-based solution for detecting Autism Spectrum Disorder (ASD) from social media data and biomedical images. They use pretrained models like Xception, VGG19, and NASNETMobile for classification tasks. Their study shows the Xception model achieving the highest accuracy of 91%, followed by VGG19 (80%) and NASNETMobile (78%). This approach aids in detecting ASD based on facial features, utilizing deep learning techniques and transfer learning.

(Opara et al., 2019) [195] propose HTMLPhish, a deep learning-based data-

driven end-to-end automatic phishing web page classification approach. HTMLPhish employs CNNs to learn the semantic dependencies in the textual contents of HTML documents. The CNNs learn appropriate feature representations from HTML document embeddings without extensive manual feature engineering. By concatenating word and character embeddings, the model effectively manages new features and ensures easy extrapolation to test data. Comprehensive experiments on a dataset of over 50,000 HTML documents yield a 93% accuracy and true positive rate. HTMLPhish is language-independent and operates as a client-side strategy, detecting phishing regardless of textual language.

2.4.3 RNN-based Approaches

Recurrent Neural Networks (RNNs) have emerged as influential architectures in text classification (TC), renowned for their ability to capture long-range dependencies and latent relationships in sequential data [166, 196]. RNNs process sentences represented as sequences of word embeddings, with each word sequentially fed into the model. Their structure, sharing parameters across different parts, allows for capturing historical and location information among words, making them suitable for various TC tasks.

Long Short-Term Memory networks (LSTMs), a popular RNN variant, effectively address the gradient vanishing or exploding issues typical in standard RNNs [196]. LSTMs are designed with a cell to remember values over arbitrary time intervals and three gates (input, forget, and output) to control information flow. This architecture enhances the connection among context feature words and filters out irrelevant information, boosting the overall classification ability. (Jain et al., 2018) [197], they investigate spam classification using a deep learning approach, specifically LSTM. This method, which learns abstract features instead of relying on hand-crafted ones, is applied to text converted into semantic word vectors using tools like Word2Vec [6], WordNet [198], and ConceptNet [199]. Their approach is benchmarked against traditional classifiers like SVM, Naïve Bayes, ANN, KNN, and Ran-

dom Forest, using the SMS Spam Collection and Twitter datasets. The results, evaluated on accuracy and F-score, reveal that LSTM outperforms traditional machine learning methods in detecting spam.

(Tang et al., 2022) [200] propose a deep learning-based framework for detecting phishing websites, implemented as a browser plug-in. This plug-in can determine phishing risks in real-time, combining strategies like whitelist filtering, blacklist interception, and machine learning (ML) prediction. The ML prediction module, utilizing several datasets, found that the RNN-GRU model achieved the highest accuracy of 99.18%, demonstrating the framework's feasibility.

Several LSTM-based models have been introduced for TC. Tree-LSTM [201] extends the LSTM architecture to tree structures, offering a more apt representation for phrases. TopicRNN [202] combines latent topic models with RNNs to better handle long-range dependencies. Universal Language Model Fine-tuning (ULMFiT) [66] employs discriminative fine-tuning on an LSTM network, optimizing it with different learning rates across layers. The Disconnected Recurrent Neural Network (DRNN) [203] enhances RNNs with position invariance, a trait typical of CNNs, using gated recurrent units (GRU) [204].

Bidirectionality in RNNs, especially LSTMs, has shown significant benefits [205]. Models like ELMo [62, 206] use bidirectional LSTMs, marking early advancements in the development of contextualized word embeddings. These approaches have consistently outperformed baselines in various TC tasks, particularly sentiment analysis.

In specialized applications, RNNs have been adapted for more complex tasks. For instance, in sentiment classification, the RNN-Capsule model [207] uses a simple capsule structure to capture feature relationships. Virtual Adversarial Training (VAT) [171] applied to RNNs improves word embedding quality and training robustness. In Natural Language Inference (NLI), the Bilateral Multi-Perspective Matching (BiMPM) model [208] employs a BiLSTM encoder for sentence encoding and matching in two directions.

2.4.4 Transformer-based Approaches

(Yamoun et al., 2023) [209] explore deep learning techniques, particularly transformers and attention mechanisms, for web content classification with a focus on detecting pornographic websites. They address challenges such as the diverse nature of web content and the existence of implicit adult content. The authors propose an approach combining attention mechanisms and transformers to classify web pages and entire websites, emphasizing the importance of considering multiple pages from a single domain. Their experiments demonstrate an accuracy rate of 91.59% on a hand-labeled test set, highlighting the benefits of employing deep learning models for web content classification, especially in porn detection.

In another study, (Yamoun et al., 2022) [210] discuss the application of the RoBERTa transformer model in website content-based classification. They compare the effectiveness of RoBERTa embeddings with traditional TF-IDF features and explore various classification approaches, including mono-multi-classification and binary classifications using the "one vs. all" strategy. The research shows that RoBERTa embeddings significantly outperform TF-IDF features, particularly in binary classifications. A 3-layer fully connected neural network is found to outperform traditional machine learning classifiers, underscoring the potential of deep learning, specifically RoBERTa, in website content-based classification.

(Demirkıran et al., 2020) [211] present a method for website classification using deep learning models, specifically focusing on a fine-tuned BERT model and LSTM classifiers with GloVe embeddings. The study utilizes the 5000best.com dataset, comprising URLs, categories, and textual descriptions of 5000 websites across 32 categories, with a notable imbalance towards the Web category. The methodology includes rigorous data preprocessing steps such as text extraction, lowercase conversion, removal of stop words, punctuation, and irrelevant symbols, followed by lemmatization. Two LSTM models are employed: one using 300-dimensional GloVe embeddings and another com-

bining 100-dimensional GloVe word embeddings with 30-dimensional character embeddings. The fine-tuned BERT model leverages its bidirectional transformer architecture to generate contextual word representations and incorporates a dense layer for classification. Results indicate that the BERT model achieves 67.81% accuracy, outperforming the LSTM models, which achieve 60.12% and 63.15% accuracy, respectively. This approach enhances the classification of web content, particularly in identifying harmful information, and demonstrates the efficacy of advanced NLP techniques in web filtering and content moderation.

In conclusion, both traditional machine learning techniques and deep learning techniques have demonstrated their effectiveness in web page classification. However, the choice of technique depends on the specific requirements and constraints of the classification task.

2.5 Hybrid Approaches

Hybrid approaches that combine traditional machine learning and deep learning have been explored to enhance web page classification. (Matosevic et al., 2021) [212] used machine learning algorithms based on expert knowledge to classify web pages into three predefined classes according to the degree of content adjustment to SEO recommendations. (Sebők and Kacsuk, 2020) [213] presented a machine learning-based solution for matching the performance of the gold standard of double-blind human coding in content analysis for comparative politics. (Raj et al., 2021) [214] proposed a neural network-based framework with parameter optimization and an algorithmic comparative study of eleven classification methods, including four traditional machine learning and seven shallow neural networks, on two real-world cyberbullying datasets.

(Özel, 2011a) [215] evaluated three classifiers: Decision Tree (J48), Naïve Bayes Multinomial (NBM), and KNN. The results showed that the best classifier varied depending on the specific class being considered. For the

Conference class, all three classifiers achieved high accuracy, with NBM and J48 reaching a maximum accuracy of 93%. However, for the Course and Student classes, the KNN classifier outperformed the others, achieving an accuracy of 92% and 83%, respectively, when using the GA-selected features. This highlights the importance of selecting an appropriate classifier based on the specific classification task. Also, (Özel, 2011b) [216] explores various features and classifiers for web page classification, including stemmed terms, HTML tags, and classifiers such as NB, KNN, decision trees, SVM, and rule induction algorithms.

(Yan et al., 2019) [217] presented a method for identifying malicious URLs using deep learning techniques to overcome the limitations of traditional machine learning methods. The proposed approach employs Stacked Denoising Autoencoders (SdA) to automatically extract high-level features from URLs, followed by a logistic regression classifier to distinguish between malicious and benign URLs. The model achieved an accuracy of 98.52% and a micro-averaged F1 score of 0.98 on a dataset of approximately 4 million URLs, demonstrating significant improvement over existing methods.

(Lopez et al., 2007) [218] proposed an unsupervised technique for term selection in information retrieval, combining the Transition Point (TP) method with bigram enrichment. This method reduces the vocabulary size of a text corpus while maintaining or improving retrieval performance. The study concluded that the combined use of TP and entropy methods enhances term selection effectiveness, offering a balance between precision and recall in information retrieval tasks.

(Khade et al., 2012) [219] introduced an alternative method for web page categorization, employing a hybrid neural network architecture. They represented a web page as a feature vector, with weights determined by term frequency and sentence significance. Principal Component Analysis (PCA) was used to select relevant features, and the output of PCA was fed into the Self-Organizing Feature Map (SOFM) for classification. This approach showed substantial improvement in classification performance compared to

KNN and NB methods.

(Espinosa et al., 2021) [220] developed an approach to classify web content using visual information from rendered homepage snapshots. Full-page images of websites were segmented into smaller sub-images using sliding windows, processed by a pre-trained deep learning model to extract feature vectors, which were then used to train an Extreme Learning Machine (ELM) classifier. The ELM model achieved high accuracy and efficiency, surpassing traditional deep learning models in speed and the ability to learn from small datasets.

(Wai et al., 2018) [221] presented an ontology-based web page classification system utilizing an enhanced C4.5 DT algorithm and a NB classifier to improve accuracy. The system incorporates semantic technology through ontology to store concepts for each word, enhancing the classification process. The enhanced C4.5 algorithm calculates normalized information gain by considering both original and semantic class labels, while the NB classifier addresses unresolved classifications by the decision tree. The system demonstrated high accuracy rates of 92% for 150 web pages and 92.5% for 200 web pages, proving its effectiveness in supporting tasks such as maintaining web directories and focused crawling.

(Rehman et al., 2019) [84] propose a hybrid deep learning model combining CNN and LSTM networks for sentiment analysis of movie reviews. The study utilizes CNN to extract high-level features and LSTM to capture long-term dependencies between word sequences, with word embeddings generated using the Word2Vec method to translate text into vector representations that capture semantic meanings. The model applies convolution and global max-pooling layers followed by LSTM layers, incorporating dropout technology, normalization, and a rectified linear unit (ReLU) to enhance accuracy. Experimental results on the IMDB and Amazon movie review datasets show that the hybrid CNN-LSTM model significantly outperforms traditional machine learning techniques like SVM and NB, as well as individual CNN and LSTM models, achieving an accuracy of 91% and improving the f-measure

score by 4-8% compared to using CNN or LSTM alone. These results highlight the hybrid model's efficacy in sentiment analysis, offering better classification accuracy and effectiveness than traditional approaches.

Table 2.7: Summary of DL and Hybrid Approaches

Approach	WS/WP	Dataset	TR	Classifier	Results
Yamoun et al. (2023) [209]	Website	Custom dataset	Transformer-based embeddings	BERT, RoBERTa	BERT: F1: 0.92, RoBERTa: F1: 0.93
Yamoun et al. (2022) [210]	Website	Custom dataset	Transformer-based embeddings, attention mechanism	Transformer	Accuracy: 91.59%
Tang et al. (2022) [200]	Website	KPT-12 Dataset	HTML Features, CNN	Logistic Regression, SVM, Random Forest, RNN	RNN-GRU: 99.18% accuracy, Random Forest: 0.0047% FPR
Aris et al. (2022) [222]	Web Page	Custom dataset	TF-IDF, Word2Vec	CNN	Accuracy: 91%, Precision: 0.89, Recall: 0.88
Alsaade et al. (2022) [194]	Web Page	Kaggle (2,940 face images)	Facial features, CNNs, Transfer Learning (Xception, VGG19, NASNETMobile)	Xception, VGG19, NASNETMobile	Xception: 91% accuracy, VGG19: 80%, NASNETMobile: 78%

Approach	WS/WP	Dataset	TR	Classifier	Results
Apandi et al. (2021) [223]	Web Page	Custom dataset	BERT embeddings	SVM, Decision Tree	SVM: Accuracy: 90%, Decision Tree: Accuracy: 87%
Espinosa et al. (2021) [220]	Website	Custom dataset	TF-IDF, BERT embeddings	SVM, Random Forest	SVM: Accuracy: 89%, Random Forest: Accuracy: 87%
Raj et al. (2021) [214]	Website	Custom dataset	TF-IDF, LDA	SVM, Naive Bayes	SVM: Accuracy: 87%, Naive Bayes: Accuracy: 84%
Matosevic et al. (2021) [212]	Web Page	Custom dataset	TF-IDF, BERT embeddings	SVM, Decision Tree	SVM: Accuracy: 90%, Decision Tree: Accuracy: 88%
Demirkıran et al. (2020) [211]	Website	5000best.com	Text and metadata features	BERT, LSTM	BERT: Accuracy: 67.81%, LSTM (GloVe): Accuracy: 60.12%, LSTM (GloVe + Char): Accuracy: 63.15%
Nandanwar & Choudhary (2020) [190]	Web Page	Custom dataset	Visual features, TF-IDF	CNN	Accuracy: 86%

Approach	WS/WP	Dataset	TR	Classifier	Results
Sebők et al. (2020) [213]	Web Page	Political content (newspapers)	TF-IDF	SVM, Bagging-Type Ensemble	Accuracy: 85%, F1: 0.83
Yan et al. (2019) [217]	Website	Custom dataset	URL-based features, content features	SVM, Random Forest	SVM: Accuracy: 90%, Random Forest: Accuracy: 88%
Rehman et al. (2019) [84]	Website	IMDB (40,000 reviews), Amazon (2000 reviews)	Word2Vec	Hybrid CNN-LSTM, CNN, LSTM	Hybrid CNN-LSTM: 91% accuracy
Opara et al. (2019) [195]	Web Page	HTML documents (47,000 legitimate, 4,700 phishing)	Word and Character Embeddings	CNN	HTMLPhish-Full: 0.98 accuracy
Lopez et al. (2019) [188]	Web Page	Extended Web Page Classification Dataset (365 web pages, 4027 images)	VGG16, Fisher Vector, VLAD	L-SVM, Logistic Regression, k-NN	fc2: 91.24%-98.29% accuracy, fc1: 90.90%-97.95%
Wang et al. (2018) [192]	Web Page	AIC Dataset (150,000 images), NPDI Dataset	ResNet50, GcNet, SpNet	LocoaNet	LocoaNet: 96.3% accuracy

Approach	WS/WP	Dataset	TR	Classifier	Results
Wai et al. (2018) [221]	Web Page	Custom dataset (HTML documents in computer science domain)	Semantic features, ontology	Enhanced C4.5, Naive Bayes	C4.5: Accuracy: 92%, NB: Accuracy: 92.5%
Gu et al. (2016) [157]	Web Page	Custom dataset	Content features, link features	SVM, Decision Tree	SVM: Accuracy: 89%, F1: 0.87, Decision Tree: Accuracy: 85%, F1: 0.83
Moustafa (2015) [193]	Website	NPDI Pornography Database	Deep learning features	AlexNet, GoogLeNet	AlexNet: Accuracy: 92.01%, GoogLeNet: Accuracy: 93.7%
Khade et al. (2012) [219]	Web Page	Custom dataset (web pages)	Object-based features, visual features	RBF Network, Random Subspace	Accuracy: 83%, Precision: 0.80, Recall: 1.0 (RBF Network)
Özel (2011) [215]	Web Page	Custom dataset	Genetic algorithm-based features	Genetic Algorithm, SVM	Genetic Algorithm: Accuracy: 88%, SVM: Accuracy: 85%

Approach	WS/WP	Dataset	TR	Classifier	Results
Özel (2011) [216]	Web Page	Conference, Course, and Student datasets from WebKB	HTML tags and terms	Genetic Algorithm, Naive Bayes, k-NN	Genetic Algorithm: Accuracy: 95%, Naive Bayes: Accuracy: 89%, k-NN: Accuracy: 86%
Lopez et al. (2007) [218]	Web Page	Custom dataset	Competitive neural network features	Neural Network	Accuracy: 86%, Precision: 0.84

Table 2.7 summarizes various deep learning and hybrid approaches for web page classification. It includes diverse models such as Transformers, RNNs, CNNs, and hybrid models combining traditional and deep learning techniques.

2.6 Conclusion

This literature review provides a comprehensive exploration of the various methodologies used in website and web page classification. It offers insights into the evolution of the field, from the initial rule-based and statistical methods to the more advanced deep learning methods in use today. The review highlights the continuous innovation and improvement that has shaped the journey of text classification over time.

Traditional text representation methods such as Bag-of-Words (BOW) and Term Frequency-Inverse Document Frequency (TF-IDF) played a crucial role in laying the foundation for text representation. They provided initial insights into the capabilities and limitations of early text representation efforts. However, these methods also highlighted the challenges of semantic losses and high-dimensional data, which called for more sophisticated approaches.

The transition to deep learning brought about a paradigm shift in text representation. It introduced word embedding like Word2Vec, GloVe, and FastText, which greatly improved the semantic comprehension of text. These models captured complex linguistic relationships, reducing the reliance on manual feature engineering and enabling more flexible and context-aware text representations.

The review also highlighted the contributions of traditional machine learning for website and web pages classification methods like Naive Bayes, KNN, Decision Trees, and SVMs, which, despite their limitations, provided foundational insights into classification techniques. The advent of neural network

architectures, including MLPs, CNNs, RNNs, and attention mechanisms, further expanded the field, using deep patterns in data to improve accuracy and adaptability.

Hybrid and ensemble approaches emerged as solutions to combine the strengths of multiple methodologies, optimizing classification accuracy and efficiency. These approaches, including HDLTex, RMDL, and HE-AGCRCNN, represent the cutting edge in creating robust, scalable systems capable of handling the challenges of modern web content.

It is noteworthy that the majority of the existing research primarily addresses web page classification rather than the broader challenge of website classification. This distinction highlights a significant gap in the literature, indicating the need for further exploration and development of methodologies specifically tailored to website classification.

The next chapter, "WeCA: A Website Classification Approach," will introduce a collaborative system that combines the strengths of traditional and modern methodologies. This system aims to address the website classification issue and provide a robust and accurate approach that can be applied in various industries.

Chapter 3

WeCA: A Website Classification Approach

3.1 Introduction

Websites are often made up of multiple web pages with extensive text content. The website classification issue is not just a simple text classification problem due to the large volume and heterogeneity of text.

This chapter aims to introduce an approach to website classification. It includes the use of modern embedding techniques like Doc2Vec and GloVe, which offer richer semantic and contextual representations of text compared to traditional methods. It combines various machine learning classifiers to improve accuracy and efficiency. By leveraging the strengths of different classifiers, ranging from traditional models like NB and SVM to more advanced ones like BERT, we aim to create a more robust classification system.

This chapter is structured as follows: Section 1 explores collaborative classification approaches and their application to website categorization. Section 2 introduces our classification approach, which uses multiple classifiers to predict website categories based on web page content. Section 3 describes the used dataset for evaluation, the experiments conducted, and the results of using aggregation strategies like Majority Voting, Borda count, and an MLP meta-classifier. Finally, the chapter concludes with a summary of our find-

ings, a discussion of implications and challenges, and suggestions for future research on website classification.

3.2 Collaborative Classification Approaches

Classifying web content effectively is a complex challenge, and collaborative classification approaches offer a promising solution. These approaches integrate diverse classifiers to leverage their strengths and compensate for their weaknesses.

Classifier combination faces several key challenges [224, 225]. Ensuring diversity among classifiers is crucial to avoid correlated errors that can undermine performance. Determining optimal weights for combining classifiers, especially in weighted voting schemes, is complex, as incorrect weights can degrade ensemble effectiveness. The computational complexity of advanced methods like Boosting and Bagging poses significant challenges, particularly with large datasets or real-time applications. Scalability issues further limit practical applicability in big data scenarios. Extensive parameter tuning increases the complexity and time required for training, while overfitting remains a risk, particularly with neural networks. Balancing ensemble complexity with interpretability is essential, as more complex models can be difficult to understand. Adapting to dynamic changes in data distribution and handling noisy, incomplete, or imbalanced data are ongoing challenges. Finally, developing effective consensus mechanisms to resolve conflicts between classifiers is critical to ensuring optimal decision-making and overall performance [226].

To tackle these challenges, various strategies for combining classifiers are suggested [224–285] . They can be classified based on when integration takes place: input data stage, feature stage, and decision stage. At the **data stage**, raw data is merged before feature extraction and classification [286]. This method enhances the dataset by consolidating raw data from various sources but needs meticulous data preprocessing to guarantee com-

patibility. At the **feature stage**, feature vectors from multiple classifiers are concatenated, providing a more comprehensive feature set. Finally, the **decision stage** involves combining the final outputs of individual classifiers using various techniques.

In the following we provide the most important combination strategies through their different categories. In Table 3.1, we summarize the most important classifier's combination strategies, outlining their advantages and disadvantages.

3.2.1 Hard and Soft Combination Strategies

Hard-Strategy Combination

Hard-strategy combination methods rely on definitive classifier outputs that have been thresholded. A prevalent technique within this category is majority voting, where each classifier votes for a class label, and the label with the most votes is selected [260, 264]. This method includes three main variations: unanimous voting, which requires all classifiers to agree on a decision; more than half voting, where a simple majority suffices; and plurality voting, where the class with the most votes wins, irrespective of whether it has a majority. Despite its straightforwardness, majority voting does not account for the confidence levels of individual classifiers, potentially limiting its effectiveness.

Soft Combination Strategy

The soft combination strategy leverages the output scores of classifiers. Techniques such as sum, product, max, and min rules are employed in the fusion process [282, 287]. The incorporation of probabilities allows for a more nuanced combination of classifiers compared to hard-strategy methods.

Applications and Advantages

In practical applications, soft combination strategies are particularly useful in domains like facial recognition, where probabilistic scores can offer a more detailed analysis and enhance decision-making accuracy [233]. However,

hard combination strategies such as majority voting remain valuable, especially when simplicity and computational efficiency are prioritized. Weighted voting strategies have evolved from majority voting, assigning varying influence levels to classifiers based on their accuracy, thereby improving the aggregation process [225, 264]. Both hard-level and soft-level classifier combination techniques provide a versatile toolkit for enhancing classification accuracy. Soft-level combiners offer probabilistic finesse, while hard-level voting systems provide categorical clarity, contributing to a multifaceted approach in machine learning decision-making. The development of various enhanced iterations of majority and weighted-majority voting techniques is underpinned by the careful calibration of weights and the consideration of prior probabilities associated with each class.

Advanced Combination Strategies

The classifier combination process can be significantly enhanced by meticulously selecting the weights assigned to classifiers and incorporating the prior probabilities of individual classes into decision-making. (Muhlbaier et al., 2009) [253] introduced a dynamic weighted consult-and-vote system designed for the incremental learning of new classes. This method addresses the "out-voting" issue common in traditional methods when confronted with new class categories. By assessing each classifier's relative performance on training data, this system facilitates a collaborative determination of voting weights for each test instance, thereby improving adaptability and accuracy.

Further refinements to the majority voting paradigm include the divide-and-conquer strategy [277], which simplifies decision-making by breaking down the classification problem into manageable sub-tasks. A quality-based combination approach [247] prioritizes classifiers based on their reliability under specific conditions, such as image quality in facial recognition applications, underscoring the importance of contextual factors in optimizing classifier performance.

Various other classifier combination techniques have been developed to address dimensionality challenges [250], optimize decision combination [263],

and enhance accuracy in binary classification problems [259, 261, 266, 279]. These strategies highlight the diversity and adaptability of classifier combination methods in achieving maximum accuracy in complex decision-making environments [235, 240, 249, 256, 265, 268, 272, 273, 275].

3.2.2 Adaptive Combination

Integrating classifiers using adaptive techniques represents significant progress in machine learning. These techniques leverage algorithms like neural networks, genetic algorithms, and fuzzy set theory to enhance classification. Artificial Neural Networks ANNs, for instance, draw inspiration from cognitive functions and are used as foundational classifiers [264]. ANNs play a key role in combining classifiers, particularly through the Multilayer Perceptron MLP [288].

(Bogdanov et al., 2008) [255] introduced Attractor Dynamics (AD) and Classifier Masking (CM) algorithms, which mimic the central nervous system's sensory integration processes. CM, a non-neural counterpart to AD, enhances robustness by discarding erroneous outputs from compromised classifiers. The combination of ANNs and SVMs for analyzing remotely sensed data [271] highlights the efficacy of ANNs in classifier combinations. The flexibility of ANNs in classifier combination is further demonstrated in research by [239], showing improved performance across classifiers.

Adaptive methodologies like adaptive weighting and the Mixture of Experts model [278] emphasize the strength of combining classifiers. The Fuzzy Stacked Generalization method [243] uses a hierarchical architecture to combine classifier decisions, significantly improving accuracy. Techniques like fuzzy logic enhance classifier combinations in both binary and broader contexts [274, 281].

The transition from non-adaptive to adaptive classifier techniques represents a shift toward more efficient and effective integration methods. Adaptive methods [239, 243, 251, 255, 271, 278, 281] offer a strategic advantage by

ensuring higher accuracy and robustness in complex classification scenarios.

3.2.3 Advanced Ensemble Approaches

The use of ensemble-based systems and combination techniques represents a significant leap in machine learning, particularly for large-scale classification tasks. These approaches involve grouping classifiers into strategic subgroups to leverage diversity, which enhances the robustness of the ensemble and enriches the exploration of the feature space [227, 228]. Bagging and its derivatives, such as Random Forest [230, 231] and Pasting Small Votes [260], introduce randomness and diversity to the ensemble by generating bootstrapped datasets from the original data.

Boosting takes a different approach, sequentially training classifiers to correct the errors of the previous models. Adaptive Boosting AdaBoost, as a meta-algorithm, integrates weak classifier's outputs into a weighted sum for the final decision. Despite being vulnerable to noise and outliers, AdaBoost has demonstrated resilience in challenging scenarios like hybrid HMM/NN speech recognition [280]. SVM classifiers and Global AdaBoost have also been employed in feature-level combination strategies to tackle high-dimensional challenges [237].

3.2.4 Hybrid Combination Approaches

Hybrid approaches combine classifier selection and aggregation [269] to balance the precision of individual classifiers and the collective strength of the ensemble. These methods adapt well to various applications, such as vacant parking space detection [238]. The integration of confusion matrix information for classifier reconciliation [270] and the use of F-measure metrics with SVM classifiers for specific tasks like emotion recognition [229] illustrate the adaptability of these techniques across applications.

Newer strategies incorporate active learning, PCA, and graph-theoretical

clustering. Active learning algorithms autonomously select the most informative data points, significantly reducing labeling costs and enhancing classification accuracy [236, 267]. Transforming correlated classifiers into uncorrelated eigen-classifiers using PCA [248, 252] and kernel-based PCA for nonlinear dependencies [245] provide enhanced ensemble performance with fewer classifiers. Furthermore, the extraction of class boundaries and application of local linear rules [289], alongside the utilization of weighted averaging in conjunction with SVM classifiers [246, 284], have proven effective across various datasets.

Table 3.1: Summary of Classifier's Combination Strategies

Strategy	Advantages	Disadvantages
Hard Strategy	Simple implementation, no training required, effective with diverse classifiers, improves accuracy with well-estimated weights, and uses a confusion matrix for improved accuracy	Performance depends on individual classifier's accuracy, not suitable for correlated classifiers, finding optimal weights is challenging, may not reflect true confidence levels, and requires accurate confusion matrix
Soft Strategy	Handles uncertainty, improves reliability, simple implementation, handles probabilities, and improves robustness	Computationally demanding, requires accurate prior knowledge and probability estimation, sensitive to estimation errors

Strategy	Advantages	Disadvantages
Adaptive Combination	Optimizes weights dynamically, adapts to new data, handles non-linearities and uncertainty, improves robustness, and adapts to varying conditions	Computationally expensive, complex implementation, requires extensive tuning, and sensitive to incorrect confidence levels
Advanced Ensemble Approaches	Reduces variance and bias, handles complex decision boundaries, effective under noisy conditions, improves predictive performance, and reduces overfitting	Inefficient with large datasets, sensitive to noise and outliers, computationally expensive, requires careful tuning, and may overfit
Hybrid Approaches	Reduces dimensionality, improves combination efficiency, handles correlations, simplifies problem-solving, and improves robustness	May lose important information, increases complexity with subtasks, requires effective consensus mechanism, and complex implementation

3.3 Discussion

Classifier combination techniques significantly enhance predictive accuracy and robustness in website classification by leveraging the strengths of multiple classifiers. Hard strategy combinations, like majority voting and Borda count, offer simplicity and computational efficiency, making them ideal for real-time applications. Majority voting aggregates votes without considering classifier confidence, which may lead to suboptimal decisions. Borda count provides a more nuanced aggregation by ranking classes but is more complex and computationally intensive.

Soft strategy combinations use output scores and probabilities, allowing for refined aggregation of classifier outputs. These methods handle uncertainty well and incorporate classifier confidence levels, enhancing robustness and decision-making granularity. However, they require accurate probability estimation and are sensitive to errors, making them computationally demanding, especially with large datasets.

Adaptive combination strategies dynamically optimize weights and adapt to new data, offering flexibility and robustness in dynamic environments. Techniques like genetic algorithms and fuzzy logic handle uncertainty and complex systems effectively but are computationally intensive and require extensive resources and time.

Advanced ensemble approaches, such as bagging, boosting, and stacking, combine diverse models to improve predictive performance. While effective, these methods are sensitive to noisy data and computationally expensive. Neural network ensembles, like MLPs, capture complex patterns but add to computational costs.

Hybrid approaches integrate multiple strategies, leveraging their strengths to achieve superior performance. Techniques like Principal Component Analysis (PCA) reduce dimensionality, and divide and conquer simplify decision-making. However, hybrid approaches introduce additional complexity and

require careful implementation to avoid overfitting and excessive computational demands.

In the context of website classification, scalability, resource consumption, and execution time are critical factors. Hard strategies like majority voting and Borda count are computationally efficient and scale well to large datasets. However, they may not fully exploit nuanced information. Soft strategies and advanced combiners, such as stacking and hybrid approaches, offer more refined aggregation but require more computational resources. MLPs enhance the classifier combination process, capturing intricate patterns and relationships, while majority voting and Borda count provide robust, efficient aggregation methods suitable for large-scale tasks with resource constraints. By selecting and applying appropriate combination strategies, high accuracy and robust performance in website classification can be achieved.

3.4 Website Classification Approach (WeCA)

In addressing the open issue of underutilization of advanced feature extraction techniques and deep learning architectures in website classification, this chapter introduces the Website Classification Approach (WeCA) (see Figure 3.1). Traditional methods [163, 215, 216, 220, 290–299] in website classification have predominantly relied on conventional feature extraction techniques, which, despite their utility, fall short of leveraging the full potential of recent advancements in text embedding technologies. These techniques, known for their efficiency, are capable of capturing a more nuanced contextual and semantic understanding of text, thereby promising enhancements in classification accuracy. Moreover, the emergence of deep learning architectures, particularly those based on transformers, has revolutionized the analysis of unstructured data, such as text. These architectures have demonstrated superior effectiveness in various domains, yet their application in website classification remains surprisingly limited. This gap underscores a significant opportunity for innovation within the field, an opportunity that WeCA

seeks to exploit.

Additionally, WeCA addresses another frequently overlooked aspect of website classification: the utilization of metadata. Metadata, often a rich source of informative cues, can markedly improve classification outcomes but has been notably underutilized in existing research endeavors. Therefore, this chapter concentrates on harnessing advanced feature extraction methods, deploying deep learning architectures, and incorporating metadata analysis into the process of website classification. By adopting this comprehensive approach, we anticipate not only to enhance the precision and efficiency of classifiers but also to contribute substantively to the evolution of website classification methodologies.

This work introduces the application of the Doc2Vec model, trained on a custom compiled dataset, marking a significant advancement in text representation techniques within the domain of website classification. Uniquely, it leverages the comprehensive capabilities of the BERT model to analyze the full textual content of web pages, extending beyond the limited scope of using only web page descriptions, a methodological enhancement over the approaches documented in previous works such as those cited by [211].

The method presented in this chapter involves developing and implementing an advanced aggregation method tailored specifically for website classification. This approach strategically analyzes text from index pages and adjacent pages within the same domain to capture a comprehensive context, an area not yet fully explored in existing literature. Therefore, the research introduces an aggregation technique that systematically incorporates contextual information from these surrounding web pages, enriching the classification process. By focusing on deep learning models for thorough text analysis and innovative aggregation methods, this work contributes significantly to the fields of machine learning and website classification.

By pushing the boundaries of existing methods and introducing approaches to data representation and aggregation, this work sets a new benchmark for future research in website classification. It opens up new avenues for

exploration and sets the stage for further advancements in the application of machine learning techniques to the complex task of classifying websites, thereby enhancing the accuracy, efficiency, and depth of analytical capabilities in this increasingly important area of study.

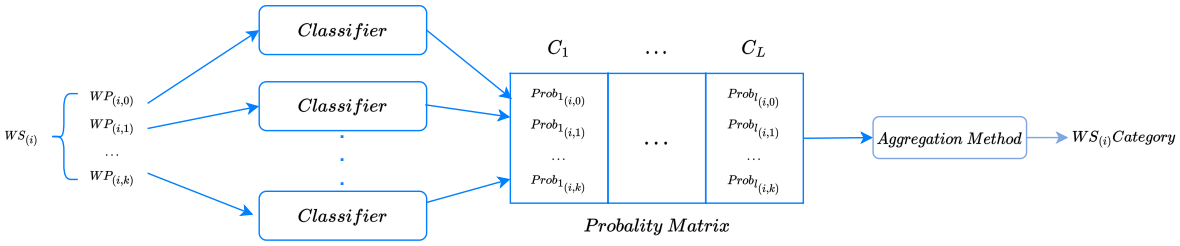


Figure 3.1: Overview of our Website classification architecture WeCA

The diagram in Figure 3.1 illustrates the architecture of our Website Classification Approach (WeCA). It delineates the flow from individual web pages to the aggregated classification of a website. For any given website $WS_{(i)}$ in the dataset, indexed as the i^{th} entry, each web page $WP_{(i,k)}$ belonging to this website is considered as an independent unit for classification. The categories into which the web pages might be classified are represented as $\{C_1, \ldots, C_L\}$. The classifiers provide a probability distribution over these categories for each web page, denoted as $[Prob_{1_{(i,0)}}, \ldots, Prob_{l_{(i,k)}}]$. These predicted probabilities form the basis for determining the final category of the website, designated as $WS_{(i)}Category$. This final prediction encapsulates the collective inference drawn from the classification results of individual web pages that comprise the website.

3.4.1 Web Page Classification

The web page classification component of WeCA is methodically structured to address the complexities inherent in analyzing the vast array of content available on the internet. This process is underpinned by a series of sequential steps, each designed with the objective of harnessing the diverse nature of web content for accurate classification. The initial stage involves data acquisition, where sophisticated web crawling technologies are deployed to amass

a comprehensive dataset that reflects the wide-ranging content found across the internet. This dataset is crucial as it forms the basis for all subsequent analyses and classification efforts.

Following data acquisition, the next step focuses on the extraction and cleaning of textual content from the web pages. This phase is critical for ensuring the quality of the data, involving the removal of irrelevant content (such as advertisements and navigational elements) and the standardization of the remaining text. This process is essential for preparing the data for detailed analysis, enabling more accurate and efficient classification.

Once the data is cleaned and prepared, it is organized into two distinct sets: one for training and the other for evaluation. This division is pivotal for the validation process, allowing for the assessment of the classification model's performance on unseen data. Such validation is essential for determining the model's generalizability and accuracy in real-world applications.

The comparative analysis of different modeling techniques constitutes the core of our methodology. Without getting into exhaustive details about each model, our approach encompasses the examination of classical machine learning models (such as SVM and NB) alongside the more advanced transformer-based BERT model. This juxtaposition allows us to evaluate the efficiency and applicability of traditional versus contemporary NLP techniques in web page classification.

This approach, spanning from data collection to model evaluation, is designed to enhance our understanding and capabilities in web page classification. It lays the foundation for the subsequent website classification phase in WeCA, providing essential insights into the potential and limitations of current classification techniques.

In conclusion, the web page classification phase is a crucial precursor to the more comprehensive task of website classification within WeCA. By systematically categorizing individual web pages, we not only refine our methodological approach but also establish a robust foundation for the classification

of entire websites. This progression from the micro (web page) to the macro (website) level of classification underscores our commitment to advancing content classification through and scientifically rigorous methodologies.

3.4.2 Website Classification

Following the comprehensive groundwork laid in the web page classification phase, our Website Classification Approach (WeCA) progresses to the pivotal task of website classification. This section focuses into the intricate process of extrapolating from the individual classifications of web pages to derive a coherent and accurate classification for entire websites. Recognizing the complexity of this task, we adopt a strategic approach that involves benchmarking and reviewing multiple collaborative classification strategies to identify the most effective methodology.

The transition from classifying individual web pages to categorizing entire websites necessitates a sophisticated framework that can accommodate the diverse and multifaceted nature of web content. In this context, collaborative classification approaches emerge as potent methodologies, enabling the synthesis of disparate classification outcomes into a unified website category. This process conceptualizes each web page as a contributing classifier, where its classification output is viewed as a vote or input towards determining the overarching website category.

Collaborative Approach for Website Classification

At this step, the approach addresses the variability and importance of different web pages within a website while effectively tackling challenges like ambiguity, noise, and the evolving nature of online content. Aggregating the results from multiple page-level classifiers enhances the robustness, reliability, and adaptability of our classification system, providing a more accurate categorization of websites despite the complexities of their content. This strategy is crucial for overcoming the inherent challenges in website classification, offering a nuanced solution that leverages the combined strengths of

individual classifiers.

Classifiers Combiner Choice In this work, we specifically tested Majority Voting, Borda Count, and an MLP as classifier combiners due to their advantages that align with the objectives of the website classification task. Majority Voting was chosen for its simplicity and effectiveness in combining diverse classifiers without the need for extensive parameter tuning, making it a reliable baseline method. Borda Count was selected for its ability to handle ranked preferences, which provides a more nuanced aggregation compared to simple majority voting, especially useful in multi-class classification scenarios.

Other classifier combiners, such as Adaptive Combinations, Advanced Ensemble Approaches like Bagging and Boosting, and Hybrid Approaches, were not chosen primarily due to their increased computational complexity and the requirement for extensive parameter tuning. These methods, while potentially offering marginal improvements in performance, involve higher resource consumption and implementation complexity, which may not be justified given the objectives and constraints of this study. By focusing on Majority Voting, Borda Count, and MLP, we aimed to balance performance with practical feasibility, ensuring robust and efficient results.

Application to Web Pages In the MLP-based combination, we consider each web page as an individual classifier of its website. This process involves generating a probability matrix for each website, where each row represents a web page and each column corresponds to the probability of that page belonging to a specific category. Given the inherent variability in the number of web pages per website, a crucial preprocessing step standardizes input vectors for the MLP. This is achieved by aggregating the probability vectors of all web pages belonging to the same website into a single vector, using the mean of these probabilities. This step ensures uniformity in the input vectors fed into the MLP, compensating for the differing cardinalities of web pages across websites.

Upon completion of training, the final categorization of each website is determined by examining the MLP's output probability vector for categories. The category with the highest probability within this vector is selected as the website's final classification. This approach capitalizes on the strengths of MLPs in handling complex patterns and relationships within data and provides a systematic and scalable solution for website classification amid the challenges posed by the variable composition of web pages across different websites.

For the Borda count and majority vote strategies, the process of classifying websites involves a unique approach to handling the probability distributions generated for each web page. Each web page effectively serves as a classifier for its respective website, contributing to an aggregate decision on the website's category based on the collective predictions of its pages.

In the Borda count-based strategy, the probabilities assigned to each category by a web page are interpreted as votes, with each category receiving points based on its position in a descending order of probabilities. Specifically, for each web page, categories are ranked according to their probability values, and points are assigned inversely to their ranks (e.g., the highest probability category receives the most points). The points from all web pages belonging to a website are then aggregated for each category. The category with the highest total points across all web pages is determined to be the final classification for the website. This method effectively accounts for the relative confidence of each web page's predictions, offering a nuanced aggregation mechanism that goes beyond mere majority rule, ensuring that more confidently predicted categories have a proportional impact on the final decision.

Conversely, the majority vote-based strategy adopts a more straightforward approach. Here, the category with the highest probability on each web page is considered the 'vote' from that page. The final classification of a website is then decided based on the category that receives the majority of votes from all web pages within that website. In cases where the number of web pages

varies significantly across websites, this approach provides a simple yet effective means of aggregating predictions. However, it primarily focuses on the most probable category from each page, potentially overlooking the valuable information contained in the distribution of probabilities across categories.

For both Borda count and majority vote strategies, the aggregation process is directly influenced by the inherent structure of the probability distributions generated for each web page. While the Borda count offers a method that leverages the full spectrum of probability distributions to inform the aggregation process, the majority vote strategy emphasizes simplicity and clarity in decision-making. Each strategy brings its strengths to the task of website classification, with the choice between them hinging on the specific requirements and constraints of the classification task, such as the desired balance between accuracy and interpretability, as well as the computational resources available.

In the following sections, we introduce our datasets used for the experiments, followed by the experimental procedures, obtained results, and an in-depth discussion.

3.5 Dataset

The experiments in this chapter relies on a subset of the Olfeo dataset, encompassing web pages across 10 categories, with the scope of the investigation restricted to English-language websites. The data collection process employs a custom crawler designed to navigate not only the target websites but also their adjacent web pages within the same domain. This approach resulted in the compilation of a dataset comprising 96741 web pages from 3125 unique domains. The categorical distribution of these domains, as shown in Figure 3.2, underscores the presence of imbalance across different categories. Textual content was extracted from the raw HTML using specified tags, similar to the method described by [290], with an additional focus on metadata tags that often include concise texts like titles and descriptions, varying in

length from 10 to nearly 100 tokens.

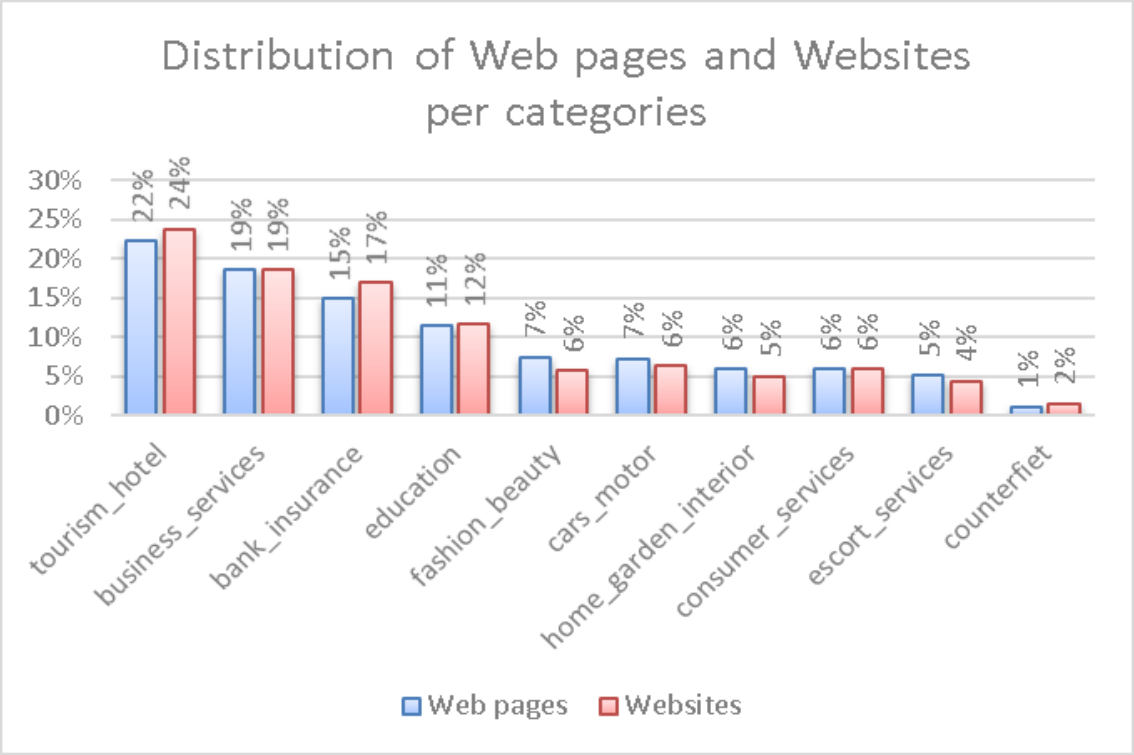


Figure 3.2: Total Web Pages per Category

The approach in this chapter encompasses two primary experimental scenarios aimed at evaluating the efficacy of various automatic website classification approaches. The datasets in question are tailored for website classification tasks and present distinct textual characteristics that offer varied challenges and insights into the classifier's performances.

The **MDO dataset** focuses exclusively on metadata, such as titles, descriptions, and keywords associated with websites. The purpose of this dataset is to analyze how well classifiers perform when they're limited to the succinct and often keyword-rich information typically used for summarization and quick reference. At depth 0, classifiers are evaluated solely on their ability to leverage this concentrated form of data to make accurate classifications. When the scenario shifts to depth 1, the classifier's performances are re-evaluated to understand the influence of neighboring web page context, this includes analyzing whether additional, potentially less structured content can aid or impede the classification accuracy when paired with the metadata.

Figure3.3 presents the distribution of tokens in the MDO dataset. Figure3.4 presents the distribution of tokens per web pages for the whole dataset at depth 1.

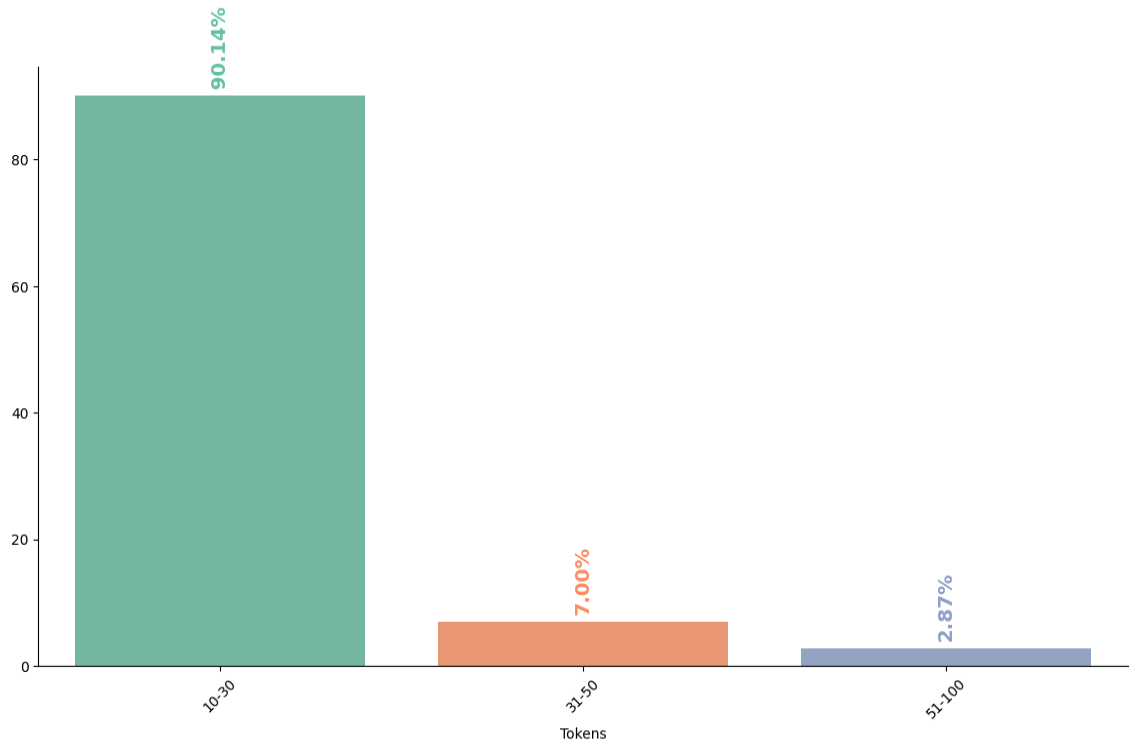


Figure 3.3: Tokens Distribution on the MDO Dataset at depth 1

The **CDO dataset** contains content-data only, sourced directly from the body text of websites. This dataset aims to simulate a scenario where classifiers have access to full, unstructured textual content, providing a more verbose and comprehensive view of a website's thematic elements. At depth 0, the classifiers' ability to discern and categorize websites based on this extensive information is put to the test. When the analysis moves to depth 1, it examines how classifiers incorporate neighboring content from linked pages, providing a wider context that could potentially enrich the classifiers' understanding and improve their classification capabilities.

Lastly, the **MCD dataset** is a fusion of the previous two, combining metadata with content data to present a more holistic dataset that represents the full spectrum of textual information a website can offer. The purpose here is to evaluate the classifiers in a scenario that closely mimics real-world conditions, where both concise metadata and elaborate content are used in tandem

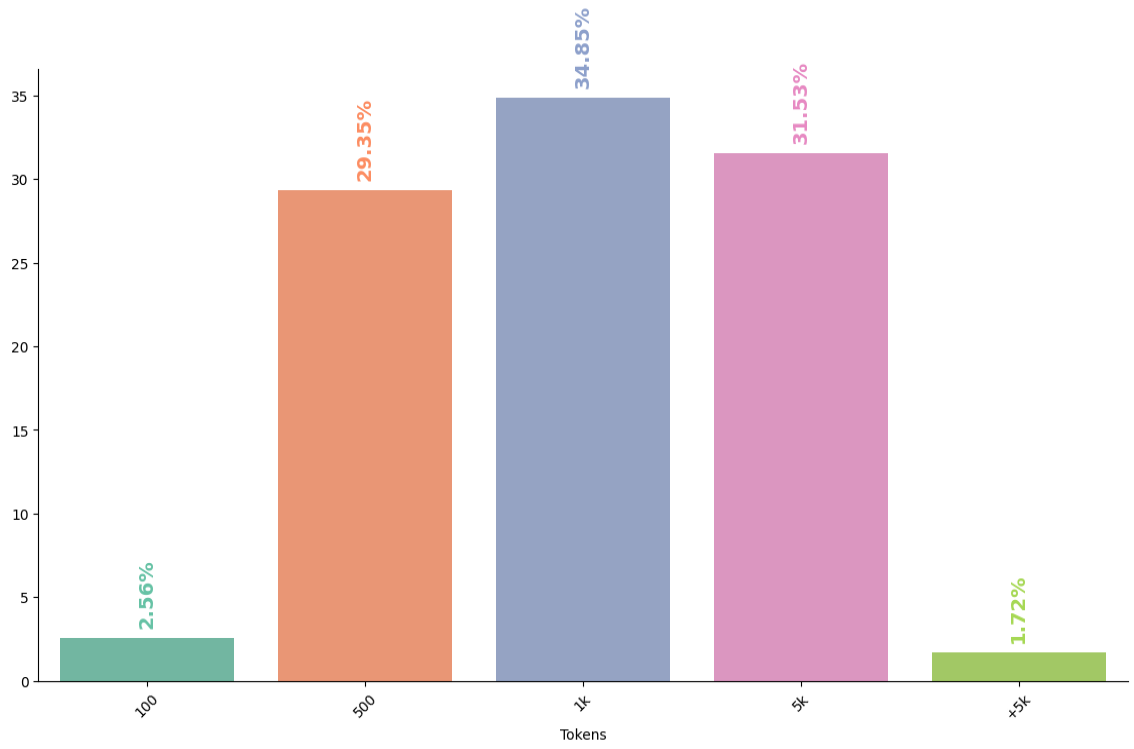


Figure 3.4: Tokens Distribution per Web Pages at depth 1

to understand and categorize websites. The meta data are concatenated at the top of each web page content with respect to their web pages.

It is noteworthy that at depth 1, the index web page is considered as any other web pages since we shuffle the data before training/fine-tuning models.

In the following we dive into the experimental protocol and details of our implementations en running environment. We follow up by deep and detailed discussion of the obtained results.

3.6 Experiments

Our experimental protocol is designed to dissect and evaluate the efficiency of various website classification approaches within the realm of web content analysis. Conducted in Python, this experiment relies on the robust capabilities of well-established libraries, ensuring the integrity and reproducibility of our results. We engaged the scikit-learn library's NB classifier implemen-

tation [300], renowned for its comprehensive algorithmic suite, and adopted the ThunderSVM classifier for its GPU-accelerated performance, following the recommendations of [301].

To generate our text embeddings, we relied on the Gensim library's Doc2Vec model [302, 303], trained for 20 epochs to reach an optimal balance between precision and computational feasibility. For our GloVe embeddings, we utilized the pretrained model on the Common Crawl corpus with a vocabulary of 2.2 million terms, leveraging the higher-dimensional vectors to enrich our text analysis with deep semantic insights. For both Doc2Vec and GloVe the vectors dimension is 300.

The intricate architectures of the CNN is based on [9] and [304]. Long Short-Term Memory (LSTM) networks were constructed within the PyTorch framework [305] based on the work of [306]. In the embedding layer we used the same previous GloVe embeddings, thus embedding contextual awareness within the classification mechanism. For fine-tuning the BERT model, we capitalized on the ktrain library (an abstraction over Keras), aligning with [307] approach to model training.

Within this experimental section, we introduce the WeCA, an approach formulated to assess the accuracy of website classification. WeCA is a composite approach encompassing various machine learning classifiers, such as SVM and NB, as well as advanced DL models, including CNN and LSTM. Each classifier is dissected across multiple web-oriented datasets—namely, MDO, CDO, and MCD—crafted to capture the multifaceted nature of web content. The unique facet of this work is its exploration of classifier efficiency at two distinct levels of data depth: the depth 0 evaluation serves as the benchmark, measuring each model's capacity to navigate the elemental aspects of web content based only on the index web page of each website. In contrast, the depth 1 examination goes deeper, revealing the potential enhancements or detriments that additional context may contribute to classification precision. Notably, this experiments also contemplated an excursion into the analysis at depth 2 but concluded that such an extension does not significantly am-

plify classifier efficacy, thus containing our inquiry within the limits of depth 1.

The data is partitioned, dedicating 70% to training and the remaining 30% to testing, ensuring a substantial data volume for model refinement while maintaining a robust set for validation. It is noteworthy that during the depth 0 phase, the need for aggregation is obviated due to the singular focus on index web pages. However, at depth 1, this process becomes essential, as the analysis expands to include the adjacent pages, thereby posing an intricate challenge of classification amidst an amalgamated data environment. This bifurcation in our protocol not only stresses the classifier's adaptability but also their resilience in the face of increasing data complexity.

The computational experiments are powered by ROMEO supercomputer center, utilizing its high-performance NVIDIA P100 NV-Link GPUs. This potent computational infrastructure is pivotal in accommodating the demanding model training and classification processes integral to the experiments.

3.6.1 Implementation Details

In the following we provide more detail about the implementation of the CNN, LSTM and MLP architectures. Furthermore, we present results of the SVM hyperparameters process.

CNN and LSTM Architectures

The implemented CNN is based on [304]. The process begins with tokenizing the input text from web pages into individual words. Each sequence is then adjusted to fixed lengths of 56, 100, and 128 tokens through truncation or padding to ensure uniform input dimensions. Tokens are mapped to 300-dimensional GloVe word embeddings, creating embedding matrices of 56x300, 100x300, and 128x300. Multiple 1D convolutional filters of various sizes (e.g., 3, 4, 5) are applied to the embeddings to capture different n-gram features. These filters are followed by "ReLU" activation functions

to introduce non-linearity. The convolutional layer outputs are then subjected to 1-max pooling, which extracts the most significant feature from each feature map, reducing dimensionality while retaining essential information. The pooled outputs are concatenated into a single feature vector, and dropout regularization is applied to prevent overfitting. This vector is passed through a fully connected layer, and a softmax activation function produces classification probabilities for each category.

The LSTM implementation is based on the work of [306]. The implementation begins with tokenizing the input text and adjusting sequences to fixed lengths of 56, 100, and 128 tokens, ensuring uniform input dimensions through truncation or padding. Tokens are then mapped to 300-dimensional GloVe embeddings, forming sequences of 56x300, 100x300, and 128x300. The embedding matrix is processed through an LSTM layer with 128 units, designed to capture temporal dependencies within the text. The LSTM architecture includes input gates, forget gates, and output gates to manage the flow of information effectively. Dropout regularization with a rate of 0.5 is applied to the LSTM output to prevent overfitting and enhance generalization. The output from the LSTM layer is passed through a fully connected layer with 128 units and a "Tanh" activation function, which helps in learning complex relationships within the data. Finally, a softmax activation function is applied to produce classification probabilities for each category.

In both CNN and LSTM implementations, the sequence length of 128 tokens proved to be the most effective. This length enables the models to capture sufficient context from the text while maintaining computational efficiency, leading to improved classification accuracy.

MLP

We aggregate the probability vectors from different classifiers using several hard methods, including sum, max, and median. After thorough evaluation, we find that the mean of the probability vectors provides the best results in terms of classification accuracy.

The architecture of the MLP model is designed to process these mean proba-

bility vectors and produce a final classification output. The MLP starts with an input layer consisting of 10 neurons, corresponding to the 10-dimensional mean probability vectors derived from the web pages. Following the input layer, the MLP includes two hidden layers, each with 128 neurons, using the "ReLU" activation function to capture complex patterns and interactions within the input data. To prevent overfitting, dropout regularization is applied after each hidden layer with a rate of 0.5, enhancing the model's generalization capabilities. The final layer is a dense layer with 10 neurons, using a softmax activation function to output the final classification probabilities for each category.

We conduct extensive hyperparameter optimization to determine the optimal configuration for our MLP model. The optimization process involves experimenting with different learning rate schedules and optimization strategies, including constant learning rate, constant learning rate with momentum, constant learning rate with Nesterov's momentum [308], inverse-scaling learning rate, inverse-scaling with momentum, inverse-scaling with Nesterov's momentum, and the Adam optimizer. The best results are achieved with a learning rate of 0.0001 and the Adam optimizer.

SVM Hyperparameters

One of the determining tasks when building machine learning models is hyperparameter optimization. A correct optimization of hyperparameters is directly reflected in the performance of the model. This is why hyperparameter optimization has been an active research area for several years. Fortunately, today there are several alternatives that can be followed for optimizing machine learning models such as [309, 310]. Each of these alternatives propose various optimization paradigms. Likewise, each of these optimization tools proposes a different usability approach which can become more or less flexible depending on the case. In their work [310], they introduced a new optimization framework called Optuna. The main goal is somehow to unify the optimization paradigms following an authoritative define-by-run API. Due to that, the code script written retains extreme modularity, and the user could actively compose "search spaces" for the hyperparameters.

Optuna's design aims to find a balance between the sampling algorithms (Gaussian Process, Random search, TPE...) and pruning algorithms (beta-alpha pruner, Asynchronous Successive Halving...).

In order to process our hyperparameter optimization task for the SVM, we use Optuna with a Bayesian optimizer as a sampler and the ASHA as a pruning algorithm. Through the framework we define our objective function which includes our search space of C, Gamma and the Kernel. The search space we set for these parameters was as follow: for C we took from $2^0, 2^1, ..., 2^n$ (we follow a power of 2 rules) with $n \in [0, 3]$, for Gamma we took from $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]$ and for the kernel, we set rbf and linear as the two main options. We redefined SVM algorithm in order to allow us to do a majority vote classification and cross validation score was used as the score to be maximized through the optimization function. After defining our trial number which we set at 20, we lunch the script and wait till we get the best parameters for SVM for each experimental scenario. Table 3.2 sums up all the SVM hyperparameters obtained with Optuna for each dataset.

Table 3.2: SVM Hyperparameters

Depth	Dataset	C	Gamma	Kernel
0	MCD	2	0.1	rbf
	MDO	4	0.01	rbf
	CDO	2	0.01	rbf
1	MCD	4	0.1	rbf
	MDO	2	1	rbf
	CDO	4	0.1	rbf

GloVe Embedding

When using GloVe with SVM and NB, we first tokenize each web page, preprocess and filter the tokens to remove any irrelevant or noisy data, and then map these cleaned tokens with GloVe embeddings vectors, resulting in a vector for each token. To obtain a final vector representation for each web page, we aggregate these token vectors by computing their mean. This averaged vector serves as the web page's representation, capturing the overall semantic content, which is then used as input for the SVM and NB classifiers.

3.6.2 Evaluation Metrics

In the following we present the main metrics we use to evaluate our models. We extract two base metrics from each experiment. These metrics are the accuracy and the F1-score that we describe below. Accuracy is defined as the number of true positives and true negatives divided by the number of true positives, true negatives, false positives, and false negatives :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

The F-score is a way of combining the precision and recall of the model, and it is defined as the harmonic mean of the model's precision and recall :

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}, \quad F1 = \frac{2 * Precision * Recall}{Precision + Recall}.$$

3.6.3 Web Page Classification

In the following we discuss the obtained results over the web page classification before any aggregation is applied at depth 1.

Depth 0 (Single Page Analysis)

In depth 0, where only the index page of each website was used for classification, BERT demonstrated remarkable performance, achieving accuracies of 75.03% on the MDO dataset, 80.59% on CDO, and 88.87% on MCD. Its advanced language comprehension capabilities proved effective in processing

Table 3.3: Classifiers Accuracies before Aggregation Strategies (Web Pages)

Depth	Clf	MDO	CDO	MCD
0	**BERT**	75,03 %	**80,59 %**	**88,87 %**
	SVM$_{Doc2Vec}$	**80,40 %**	76,79 %	77,02 %
	NB$_{Doc2Vec}$	72,01 %	70,11 %	73,80 %
	SVM$_{GloVe}$	**81,54 %**	77,01 %	76,90 %
	NB$_{GloVe}$	72,61 %	68,00 %	75,32 %
	LSTM	**81,53 %**	79,44 %	**84,12 %**
	CNN	**80,22 %**	79,11 %	**85,63 %**
1	**BERT**	**81,90 %**	**87,37 %**	**89,01 %**
	SVM$_{Doc2Vec}$	**82,70 %**	**83,08 %**	79,13 %
	NB$_{Doc2Vec}$	78,12 %	76,69 %	76,38 %
	SVM$_{GloVe}$	**84,20 %**	**84,33 %**	**85,02 %**
	NB$_{GloVe}$	79,00 %	78,55 %	78,63 %
	LSTM	**80,30 %**	**82,05 %**	**84,89 %**
	CNN	**83,13 %**	**85,70 %**	**86,00 %**

and classifying a single page, demonstrating a clear advantage in scenarios where detailed content analysis is essential.

The SVM classifiers, paired with either Doc2Vec or GloVe embeddings, displayed notable performance. SVM combined with Doc2Vec achieved accuracies of 80.40% on MDO, 76.79% on CDO, and 77.02% on MCD. However, SVM with GloVe performed slightly better with scores of 81.54% on MDO, 77.01% on CDO, and 76.90% on MCD. This suggests that GloVe's broader semantic features provided an edge in some datasets, allowing SVM to better capture the nuances in web page text.

For NB (NB), the results were more moderate. When using Doc2Vec, NB achieved accuracies of 72.01% on MDO, 70.11% on CDO, and 73.80% on MCD. The GloVe embeddings led to slightly better results, with scores of 72.61% on MDO, 68.00% on CDO, and 75.32% on MCD. This reflects NB's

simpler probabilistic approach, which doesn't fully leverage the richer text representations.

Deep learning models such as LSTM and CNN demonstrated high accuracy scores. The LSTM model achieved accuracies of 81.53% on MDO, 79.44% on CDO, and 84.12% on MCD, indicating its ability to handle sequences effectively. The CNN model showed similar performance, scoring 80.22% on MDO, 79.11% on CDO, and 85.63% on MCD, highlighting its capacity for capturing significant features in web pages.

Depth 1 (Single Page and Neighboring Pages Analysis)

At depth 1, where the analysis extended to include neighboring pages, BERT maintained its leading performance, increasing its scores to 81.90% on MDO, 87.37% on CDO, and 89.01% on MCD. The additional context from neighboring pages enriched BERT's understanding of the text, enhancing its classification capabilities.

SVM classifiers also showed improvements with the extra context. SVM combined with Doc2Vec saw its scores increase to 82.70% on MDO, 83.08% on CDO, and 79.13% on MCD. SVM with GloVe further improved to 84.20% on MDO, 84.33% on CDO, and 85.02% on MCD. These improvements highlight the importance of leveraging additional data to improve classification accuracy, particularly for models that rely on feature-rich embeddings.

NB classifiers with Doc2Vec showed modest improvements, reaching accuracies of 78.12% on MDO, 76.69% on CDO, and 76.38% on MCD. With GloVe embeddings, NB improved slightly to 79.00% on MDO, 78.55% on CDO, and 78.63% on MCD. Despite the gains, NB's simpler nature meant it couldn't fully utilize the embeddings, leading to less significant improvements compared to SVM.

Deep learning models also improved at depth 1, with LSTM reaching 80.30% on MDO, 82.05% on CDO, and 84.89% on MCD, showcasing its sequential data processing strengths. CNN improved to 83.13% on MDO, 85.70% on CDO, and 86.00% on MCD, emphasizing its robust feature extraction capabilities, which proved beneficial for classification.

SVM and NB Comparison with Doc2Vec and GloVe Embeddings

Comparing the SVM and NB classifiers with both Doc2Vec and GloVe embeddings reveals several insights. SVM with Doc2Vec provided consistent performance at both depth 0 and depth 1, indicating its strong classification capability across varied web page data. However, SVM with GloVe generally outperformed Doc2Vec across the datasets, suggesting that GloVe's semantic richness offers a more comprehensive understanding of web page content.

For NB, the difference between Doc2Vec and GloVe was less pronounced. With Doc2Vec, NB performed reasonably well but lagged behind SVM due to its simpler probabilistic assumptions. GloVe slightly improved NB's accuracy, but the model's inherent limitations meant it couldn't exploit the embeddings to their full potential.

In summary, while all models improved with additional context at depth 1, deep learning models like BERT and CNN exhibited the highest classification accuracy. SVM benefited from using GloVe embeddings, but NB did not exhibit as significant of an improvement, revealing the constraints of probabilistic models in web page classification.

3.6.4 Website Classification through Collaborative Approaches

In the following we discuss the obtained results over the website classification after the aggregation is applied at depth 1.

Majority Voting Based Strategy

Using majority voting for aggregation at depth 1 significantly improved classification accuracy for most classifiers. BERT's accuracy improved to 83.68% on MDO, 89.22% on CDO, and 90.70% on MCD. This improvement shows that majority voting can effectively leverage BERT's understanding of text and enrich the aggregation of predictions from multiple pages.

SVM classifiers also benefited from the majority voting aggregation. SVM with Doc2Vec saw accuracies improve to 85.06% on MDO, 84.27% on CDO, and 81.70% on MCD. With GloVe embeddings, SVM's accuracy reached

Table 3.4: Classifiers Accuracies (Majority Vote)

Depth	Clf	MDO	CDO	MCD
1	**BERT**	83,68 %	**89,22 %**	**90,70 %**
	SVM$_{Doc2Vec}$	**85,06 %**	84,27 %	81,70 %
	NB$_{Doc2Vec}$	82,00 %	80,61 %	80,22 %
	SVM$_{GloVe}$	**86,23 %**	**85,30 %**	**86,47 %**
	NB$_{GloVe}$	82,87 %	80,15 %	80,91 %
	LSTM	84,17 %	**85,11 %**	**87,91 %**
	CNN	**85,97 %**	**87,09 %**	**89,77 %**

Table 3.5: Classifiers Accuracies (Borda Count)

Depth	Clf	MDO	CDO	MCD
1	**BERT**	83,07 %	**88,94 %**	**90,10 %**
	SVM$_{Doc2Vec}$	**85,07 %**	83,67 %	80,97 %
	NB$_{Doc2Vec}$	80,77 %	78,80 %	79,02 %
	SVM$_{GloVe}$	**85,90 %**	**84,93 %**	**86,16 %**
	NB$_{GloVe}$	82,18 %	81,00 %	81,60 %
	LSTM	83,90 %	**85,50 %**	**87,12 %**
	CNN	**85,30 %**	**86,78 %**	**89,27 %**

Table 3.6: Classifiers Accuracies (Meta-Classifier Aggregation Strategy)

Depth	Clf	MDO	CDO	MCD
1	**BERT**	83,15 %	**88,91 %**	**90,33 %**
	SVM$_{Doc2Vec}$	**85,20 %**	84,57 %	81,38 %
	NB$_{Doc2Vec}$	81,91 %	80,00 %	79,98 %
	SVM$_{GloVe}$	**86,09 %**	**85,23 %**	**86,49 %**
	NB$_{GloVe}$	82,00 %	80,37 %	80,03 %
	LSTM	84,13 %	**85,70 %**	**87,54 %**
	CNN	**85,70 %**	**87,01 %**	**89,66 %**

86.23% on MDO, 85.30% on CDO, and 86.47% on MCD. These improvements highlight how the aggregation method helps SVM to utilize the richer text representations more effectively.

NB classifiers with Doc2Vec also improved with majority voting, achieving 82.00% on MDO, 80.61% on CDO, and 80.22% on MCD. With GloVe embeddings, NB's performance was 82.87% on MDO, 80.15% on CDO, and 80.91% on MCD. Despite NB's improvements, its performance still lagged behind other classifiers, reflecting its simpler classification approach.

Deep learning models also saw improved results using majority voting. LSTM reached 84.17% on MDO, 85.11% on CDO, and 87.91% on MCD, while CNN reached 85.97% on MDO, 87.09% on CDO, and 89.77% on MCD. The significant improvement in accuracy for these models illustrates the effectiveness of majority voting in combining their predictions.

Borda Count Based Strategy

The Borda count aggregation strategy also produced significant improvements at depth 1. BERT achieved accuracies of 83.07% on MDO, 88.94% on CDO, and 90.10% on MCD, demonstrating how the Borda count method complements its deep understanding of text.

SVM classifiers showed notable improvements using the Borda count strategy. SVM with Doc2Vec improved to 85.07% on MDO, 83.67% on CDO, and 80.97% on MCD. Using GloVe, SVM achieved 85.90% on MDO, 84.93% on CDO, and 86.16% on MCD. The Borda count helped SVM better interpret the rich embeddings, boosting its classification accuracy.

NB classifiers with Doc2Vec saw accuracies of 80.77% on MDO, 78.80% on CDO, and 79.02% on MCD, and with GloVe, they scored 82.18% on MDO, 81.00% on CDO, and 81.60% on MCD. Despite these improvements, NB classifiers still lagged behind SVM and deep learning models, highlighting NB's simpler capabilities.

Deep learning models also benefited from the Borda count strategy. LSTM achieved 83.90% on MDO, 85.50% on CDO, and 87.12% on MCD, while

CNN reached 85.30% on MDO, 86.78% on CDO, and 89.27% on MCD. This aggregation method effectively capitalized on the strengths of these models.

MLP Based Strategy

The meta-classifier aggregation strategy used an MLP to combine predictions, which resulted in further improvements. BERT achieved accuracies of 83.15% on MDO, 88.91% on CDO, and 90.33% on MCD, demonstrating the MLP's ability to effectively aggregate predictions.

SVM classifiers with Doc2Vec improved to 85.20% on MDO, 84.57% on CDO, and 81.38% on MCD. With GloVe, SVM improved to 86.09% on MDO, 85.23% on CDO, and 86.49% on MCD, highlighting the strength of combining the embeddings with a meta-classifier for improved predictions.

NB classifiers with Doc2Vec saw slight improvements, scoring 81.91% on MDO, 80.00% on CDO, and 79.98% on MCD. With GloVe, NB achieved 82.00% on MDO, 80.37% on CDO, and 80.03% on MCD. These results indicate that even though NB classifiers improved, the meta-classifier approach was more effective with complex models.

The deep learning models showed strong performance with the meta-classifier aggregation strategy. LSTM reached 84.13% on MDO, 85.70% on CDO, and 87.54% on MCD, while CNN achieved 85.70% on MDO, 87.01% on CDO, and 89.66% on MCD. The meta-classifier successfully integrated predictions from these models, showcasing the effectiveness of this strategy for enhancing accuracy.

Overall, aggregation strategies significantly improved classification accuracy for all models. BERT and deep learning models, in particular, exhibited the most substantial improvements. Majority voting provided a straightforward yet effective way to combine predictions, while the Borda count and meta-classifier strategies offered more nuanced methods for leveraging multiple classifiers. SVM benefited from these strategies as well, especially when using GloVe embeddings. NB classifiers, while showing some improvements, generally underperformed compared to other classifiers, highlighting their limitations in handling complex classification tasks.

3.6.5 Discussion

Comparing classification at depth 0 and depth 1 reveals the impact of leveraging aggregation strategies. At depth 0, where classification is based solely on the index page of websites, classifier performance is inherently limited. BERT, with its advanced language understanding capabilities, demonstrates decent accuracy, but it still falls short in capturing the full context of the website's content. Traditional models like SVM and Naive Bayes struggle more in this context, highlighting the challenges of relying on a single page for website classification.

In contrast, depth 1 incorporates aggregation methods, allowing classification to consider multiple web pages from each website. Majority voting, which involves combining the classification results of individual web pages, shows a noticeable improvement in accuracy across most classifiers. For example, accuracy for Naive Bayes on the MCD dataset jumps from around 72% at depth 0 to over 80% at depth 1. This improvement demonstrates the effectiveness of majority voting in boosting the classification results by leveraging the additional data from multiple pages.

Other aggregation methods, such as the Borda count, also display enhanced classification accuracy at depth 1. This method, which ranks each class based on its relative probability across web pages, further refines classification decisions compared to majority voting. However, majority voting stands out due to its simplicity and effectiveness, offering a significant boost in accuracy while maintaining computational efficiency.

BERT's accuracy is further enhanced at depth 1, reaching over 83% in some cases due to the aggregation strategies employed. This highlights how advanced models like BERT, when combined with effective aggregation methods, can significantly enhance website classification.

3.6.6 Impact of Metadata on Classification Performance

The choice of dataset plays a crucial role in determining the performance of classification models. In our experiments, three distinct datasets were used: Metadata-Only (MDO), Content-Only (CDO), and Metadata-Content (MCD). Each dataset presents unique challenges and opportunities for machine learning models, influencing the effectiveness of various classification approaches.

Metadata-Only (MDO): The MDO dataset, consisting solely of metadata, primarily includes structured data such as titles, tags, and other descriptors. This data type often provides a concise summary of the content, leading to high classification performance when the metadata accurately reflects the content. However, its limited scope can make it challenging for classifiers to handle ambiguous or less informative metadata. In this dataset, models using advanced embeddings like GloVe or Doc2Vec with Support Vector Machines (SVM) tend to perform well. The SVM with GloVe classifier achieved accuracy scores of 81.54% at depth 0 and improved further with majority voting aggregation.

Content-Only (CDO): The CDO dataset includes the textual content of web pages, offering a richer source of data compared to metadata alone. This dataset often provides more nuanced information, which is beneficial for deep learning models that can leverage semantic features from the full text. BERT, which excels at capturing deep contextual relationships, stands out with its performance on the CDO dataset. It achieves the highest accuracy among classifiers due to its ability to comprehend context deeply.

Metadata and Content (MCD): The MCD dataset combines both metadata and content, providing the most comprehensive representation of web pages. This dataset enables classifiers to leverage the complementary strengths of metadata and content for classification tasks. The integration of metadata and content provides classifiers like BERT and SVM with an enriched data

environment, leading to higher accuracy scores. For instance, the SVM classifier achieved 84.2% accuracy with GloVe embeddings at depth 1, while BERT reached up to 90.33% with aggregation strategies.

The results highlight that the type of dataset used significantly influences classifier performance. Classifiers generally perform best on the MCD dataset due to the combined strengths of metadata and content. However, the choice of classifier also matters. Deep learning models like BERT show remarkable adaptability to all datasets, while traditional models such as SVM and Naive Bayes vary in effectiveness depending on the dataset and embedding strategy used.

3.7 Conclusion

In this chapter, we introduced WeCA and demonstrated its effectiveness through rigorous experimental evaluation. WeCA was designed to elevate the accuracy and efficiency of website classification, capitalizing on advanced text representation techniques and strategic classifier aggregation. Our investigation spanned a variety of classifiers, including Support Vector Machines (SVM), Naive Bayes (NB), Convolutional Neural Networks (CNN), and Long Short-Term Memory networks (LSTM), alongside two prominent embedding techniques: the custom-trained Doc2Vec and the widely recognized pretrained GloVe embeddings. The analysis was conducted across multiple datasets to ensure a thorough evaluation of WeCA's classification capabilities.

A significant technological barrier that WeCA successfully overcame was the integration of diverse data sources for classification. The results unequivocally showed that combining metadata with content (as seen in the MCD dataset) considerably boosted classification performance across all classifiers. This integration harnessed the complementary strengths of both metadata and content, providing a more holistic view of web pages. Moreover, WeCA adeptly addressed the challenge of contextual information utilization. The

transition from utilizing solely index pages (depth 0) to incorporating information from neighboring web pages (depth 1) was shown to substantially improve classification outcomes. This emphasized the critical role of contextual and neighboring page data in enhancing the classifier's understanding and performance.

Among the findings, SVM and CNN classifiers demonstrated exceptional performance with GloVe embeddings, benefiting from GloVe's extensive linguistic coverage and semantic depth. Similarly, the custom-trained Doc2Vec embeddings presented promising outcomes, particularly with SVM, highlighting the advantages of domain-specific embeddings in refining classification accuracy. Furthermore, WeCA's employment of aggregation strategies, notably the meta-classifier approach, showcased the potential to fine-tune classifier outputs, achieving superior accuracy, especially in complex datasets like MCD at depth 1.

WeCA distinguished itself by addressing several technological barriers in website classification. It showcased adaptability in data source integration, effectively leveraged contextual information from web pages, and utilized sophisticated machine learning techniques to optimize classification outcomes. Through this approach, WeCA not only contributed to the advancement of website classification methodologies but also laid a solid foundation for future research and application in this domain.

Looking ahead, the next chapter will focus on addressing two pivotal challenges: scalability and the management of the vast amounts of text content encountered in web pages. The forthcoming discussion will outline a new approach designed to tackle these issues, aiming to further enhance the robustness and applicability of website classification frameworks. By innovating solutions to these challenges, we aim to extend the capabilities of WeCA, ensuring its scalability and efficiency in processing extensive text data, thereby solidifying its position as a leading methodology in the field of digital content classification.

Chapter 4

Text Chunking To Improve Website Classification

4.1 Introduction

The advent of transformer models has marked a revolutionary leap in the field of NLP, setting new benchmarks across a spectrum of tasks, from sentiment analysis and text summarization to language translation. Models like BERT and RoBERTa have been at the forefront, demonstrating unparalleled capabilities in understanding the nuances of human language. Their architecture, built on the self-attention mechanism, allows them to capture contextual relationships in text, regardless of the distance between words. This feature has enabled the development of systems that can comprehend and generate human-like text, opening up possibilities that were previously unattainable.

Despite their remarkable benefits, transformer models are not without limitations. One significant challenge is their handling of long text sequences. The quadratic complexity associated with their self-attention mechanism restricts the input sequence length, making the processing of extensive documents computationally expensive and memory-intensive. This limitation becomes particularly pronounced in tasks requiring the analysis of lengthy texts, such as classifying comprehensive web pages, where the essence and context might span several thousand words (see Figure 3.4). Traditional approaches to address this challenge, such as segmenting texts or employing

models designed for longer sequences like Longformer and BigBird, often lead to compromises on contextual integrity or entail increased computational demands.

In response to these challenges, this chapter introduces the Weighted Stratified Split Approach (WSSA), a novel methodology designed to circumvent the limitations posed by the fixed-length input constraint of transformer models. WSSA is based on text chunking and a weighted stratification strategy, enabling efficient processing of long documents while preserving the contextual richness necessary for high-accuracy classification tasks. This approach not only enhances the applicability of BERT and RoBERTa to longer texts but also offers a more resource-efficient alternative to models specifically engineered for extended sequences.

The chapter aims to provide a comprehensive exploration of the implementation and benefits of the WSSA in the context of website classification. It examines the theoretical underpinnings of transformer models, elucidates the challenges of processing long text sequences, and articulates how the WSSA addresses these issues. Through rigorous experimental validation, the chapter demonstrates the efficiency of WSSA in improving classification performance, offering insights that extend beyond website classification to broader NLP applications.

4.2 Transformers and Long Texts

Transformer models, renowned for their revolutionary impact on various NLP tasks, encounter inherent limitations when tasked with managing extensive textual data. This section provides an overview of various strategies developed to enhance the functionality of transformer models for handling long text sequences. By exploring these methodologies, we aim to highlight how transformers can be adapted for use in scenarios that involve extensive textual data. This exploration is intended to show the transition from the theoretical capabilities of transformer models to their practical application

in managing long texts, identifying the methods that enable these models to process significantly larger amounts of data than they were originally designed for.

4.2.1 Long Text Classification Approaches

The field of long text classification has witnessed significant advancements through various innovative approaches aimed at addressing the inherent challenges associated with processing extensive textual data [311–323]. These advancements can be broadly categorized into two main strategies: enhancements to attention mechanisms for managing long texts and developments of models for long text classification.

In the field of attention mechanism improvements, the Longformer model introduced by (Beltagy et al., 2020) [311] incorporates a "sliding window" technique, allowing for the efficient processing of sequences up to 4096 tokens while preserving global context awareness. The BigBird model proposed by (Zaheer et al., 2020) [312] employs sparse attention patterns to further extend processing capabilities to sequences of up to 8192 tokens. Through the use of such specialized mechanisms both models process extensive sequences efficiently, while significantly reducing their computation complexity. For the longformer the complexity is mathematically represented as $O(n \cdot w + n \cdot g)$, where n is the sequence length, w is the fixed window size, and g is the number of tokens with global attention. This results in a linear complexity relative to the input size when w and g are kept much smaller than n. Similarly, the computational complexity of BigBird is linear with respect to the input length, expressed as $O(n)$, under standard parameter settings. BigBird enhances the reach of each token through a triad of attention mechanisms: sliding window, global and random. This design ensures that each token can effectively access a broad context, thereby maintaining comprehensive context awareness while keeping computational demands in check.

Another notable contribution is the Transformer-XL presented in (Dai et

al., 2019) [313], which introduces a segment-level recurrence mechanism designed to capture dependencies beyond fixed context lengths. Additionally, compressive transformers, as proposed by (Rae et al., 2020) [314], utilize random projections to condense long sequences, potentially reducing the computational complexity associated with attending to extended text sequences. However, the specifics of this reduction depend on the compression function's implementation details. The BP-Transformer, presented by (Ye et al., 2019) [315], employs binary partitioning and blockwise self-attention to model long-range context efficiently, achieving a complexity of $O(k \cdot n \cdot \log(n/k))$, where k is a hyperparameter controlling attention density. (Qui et al., 2019) [316] also divide sequences into blocks to compute self-attention more effectively, although this partitioning might limit the ability to capture detailed dependencies across segments.

To address scalability, (Wang et al., 2020) [317] introduced the Linformer, which leverages a linear-complexity self-attention mechanism, significantly reducing the computational demands for long sequences from $O(n^2)$ to $O(n)$. The Reformer, developed by (Kitaev et al., 2020) [320], optimizes transformer efficiency through the use of reversible layers and sparse factorizations, bringing the complexity down to $O(n \log n)$. The Performer, proposed by (Choromanski et al., 2020) [321], rethinks the attention mechanism to lower memory usage and enhance model scalability, with computational complexity akin to that of the Linformer. (Roe et al., 2020) [319] focus on computational efficiency improvements by adopting content-based sparse attention and routing transformers that leverage sparsity patterns. Furthermore, (Kitaev et al., 2020) [318] introduce an adaptive attention span feature allowing transformers to dynamically adjust the attention window, thereby increasing efficiency in managing long-range dependencies. While these strategies present significant improvements in scalability and memory efficiency for processing longer text sequences, they also introduce new complexities and require precise tuning of model parameters.

Some other works proposed to tackle the issue differently. In the research conducted by [322, 323] addresses crucial limitations of Transformer-based

models such as BERT in handling long text sequences, which is particularly challenging in the context of extensive real-world datasets.

In [322] (Tuteja et al., 2023) investigate several strategies to adapt Transformer models for processing lengthy legal documents efficiently. Their approach involves a detailed evaluation of various Transformer architectures, including RoBERTa and Longformer, against a traditional baseline combining TF-IDF with Neural Networks. The study meticulously assesses different methods of document chunking and selection to optimize text processing. Specifically, they explore strategies like selecting the most informative sections of documents or segmenting documents into manageable chunks that are individually processed and then aggregated. Their evaluation demonstrates that the performance of advanced models varies significantly across datasets, indicating the need for tailored strategies that consider the distinctive characteristics of each dataset. This nuanced approach highlights the complexity of effectively applying Transformer models to real-world NLP tasks, particularly in the legal domain.

On the other hand, (Jaiswal et al., 2023) [323] recently proposed ChunkBERT in parallel and independently to our work which introduces a new methodology to extend BERT's processing capabilities beyond the standard 512-token limit. ChunkBERT employs a strategic chunking process where the text is divided into smaller segments, each of which is processed independently using BERT. The outputs are then combined using convolutional neural networks (CNNs) to synthesize the information from different chunks. This method significantly reduces memory usage to just 6.25% of what is typically required by BERT, without necessitating specialized hardware like custom CUDA kernels. This enhancement allows for scalable and efficient processing of extended texts. The effectiveness of ChunkBERT is rigorously tested through comprehensive benchmarks across various long-text classification tasks. The results demonstrate that ChunkBERT not only effectively manages to process long texts but does so with a high degree of efficiency, making it a viable solution for a wide range of NLP applications that involve lengthy documents.

4.2.2 Discussion

The reviewed approaches reveal the complexity and challenges involved in handling long text sequences with Transformer-based models. Both sets of studies introduce innovative methods aimed at overcoming the inherent limitations of traditional models. Techniques such as attention modifications, sequence partitioning, model efficiency improvements, and the development of specialized architectures like ChunkBERT are explored to enhance scalability, memory efficiency, and reduce computational overhead. However, these approaches also come with their own set of challenges. While they show advancements in handling extensive sequences, they often introduce increased computational complexity, require extensive resource allocation, or necessitate fine-tuned parameter adjustments to achieve optimal performance.

Furthermore, approaches that partition long texts into manageable chunks can sometimes struggle to maintain the integrity of contextual relationships across partitions. This can potentially lead to a loss in modeling capacity, especially in capturing fine-grained dependencies that are critical for understanding complex texts. Therefore, it becomes crucial to carefully weigh the trade-offs between computational efficiency, modeling capacity, and the specific demands of the task. Future research should focus on refining these techniques to enhance their adaptability and ease of implementation.

In the following section we introduce the data splitting techniques used in the ML field, we then provide a detailed description of our proposed data splitting approach and its application to our website classification approach.

4.3 Data splitting and website classification approach

In the field of machine learning and statistical analysis, the process of dividing datasets into training, validation, and testing sets is crucial for building robust and generalizable models. This subdivision ensures that models are

not only trained effectively but also validated and tested under conditions that mimic unseen real-world data. Among the various techniques employed for data splitting, non-stratified and stratified splits are particularly significant. Non-stratified splits involve random division of data without regard to the distribution of variables, while stratified splits ensure that each subset of data reflects the overall distribution of key variables, especially the target or outcome variable. This section explores the methodologies and implications of these splitting strategies, providing a foundation for understanding their application and importance in machine learning workflows.

In this section, we present the basic split method that will be used as a baseline to compare with our weighted stratified split approach and then we follow up with our method for website classification.

4.3.1 Basic Split (BS)

There are two types of basic data splitting [324–327]. The first type is a basic split with non-stratified data and the second type is with stratified data. The main difference between the two lies in how the subsets are created and whether they preserve the distribution of categories in the original dataset.

Basic Non-Stratified Split (BNSS)

A **non-stratified split**, commonly referred to as a random split, is a method used to divide a dataset into multiple subsets, such as training and testing sets, without taking into account the distribution of target variables or any specific features within the data. This approach is widely used in machine learning and statistical modeling, primarily because of its simplicity and straightforward implementation [324–327].

In a non-stratified split, each data point x_i in the dataset D has an equal probability of being assigned to either the training set D_{train} or the testing set D_{test}. Typically, this is achieved using a uniform random distribution. The selection process can be mathematically described by a random variable

Z_i for each data point x_i, where Z_i follows a Bernoulli distribution :

$$Z_i \sim \text{Bernoulli}(p). \tag{4.1}$$

Here, p represents the probability of a data point being included in the training set. The value of p is often set based on the desired split ratio; for example, $p = 0.7$ for a 70% training and 30% testing split. Therefore, the process for each data point x_i can be defined as :

$$x_i \in \begin{cases} D_{\text{train}} & \text{if } \ Z_i = 1, \\ D_{\text{test}} & \text{if } \ Z_i = 0. \end{cases} \tag{4.2}$$

While non-stratified splitting is straightforward and requires minimal computation, it does not guarantee that the training and testing sets will be representative of the overall dataset, particularly in cases where the data contains imbalanced classes or skewed distributions. This can lead to models that are biased or perform poorly on unseen data due to overfitting or underfitting.

In practice, non-stratified splitting is suitable for large datasets with a relatively uniform distribution of features. However, for datasets with significant imbalances or when the distribution of the target variable is crucial for predictive accuracy, stratified splitting is generally recommended to ensure that each class is adequately represented in both training and testing subsets.

This approach underscores the importance of understanding the characteristics of the dataset when choosing a splitting strategy, as the choice can significantly impact the performance and generalizability of the resulting models.

Basic Stratified Split (BSS)

A **stratified split** is a method used in statistics and machine learning to divide a dataset into subsets (such as training and testing sets) while ensuring that each subset is representative of the entire dataset in terms of key characteristics, typically the distribution of the target or outcome variable. This technique is particularly useful when the dataset has imbalanced classes or

categories that need to be equally represented in each split to prevent biased or skewed model training results [324–327].

Consider a dataset D consisting of N samples. Let y_i be the target variable for each sample x_i, for $i = 1, 2, \ldots, N$. In a stratified split, the dataset is divided into subsets such as D_{train} and D_{test} in such a way that the proportion of each category of y in D is approximately the same in these subsets.

Let C_k be the set of indices of samples belonging to category k in the dataset, where k is one of the possible classes of the target variable y. The proportion of the dataset D that belongs to category k can be calculated as

$$p_k = \frac{|C_k|}{N}, \tag{4.3}$$

where $|C_k|$ is the cardinality of set C_k, i.e., the number of samples belonging to category k.

In a stratified split, the objective is to ensure that each subset of the dataset, for example, D_{train} and D_{test}, maintains this proportion p_k for each category k. Mathematically, if $|D_{\text{train}}| = n$ and $|D_{\text{test}}| = N - n$, the number of samples from category k in D_{train} should be approximately

$$|C_{k,\text{train}}| \approx n \cdot p_k. \tag{4.4}$$

Similarly, the number of samples from category k in D_{test} should be approximately

$$|C_{k,\text{test}}| \approx (N - n) \cdot p_k. \tag{4.5}$$

Stratified sampling ensures that each training and testing set is a good representative of the overall population, particularly important in scenarios where certain classes are underrepresented. This method helps in achieving more reliable and generalizable training outcomes, especially for classification problems where class imbalance could significantly skew the training process.

However, implementing a stratified split requires careful handling of the data to maintain these proportions, especially when dealing with multiple categorical variables or when classes are heavily imbalanced. Additionally, while stratified splits help in maintaining statistical properties across splits, they

might complicate the sampling process, requiring more sophisticated algorithms to ensure the splits are done correctly.

4.3.2 Weighted Stratified Split Approach (WSSA)

The traditional data splitting technique, commonly known as the BSS, initially guided our experiments by categorizing and distributing web pages based on their respective categories. However, this method soon revealed significant imbalances, disproportionately favoring larger websites in the training set and smaller websites in the testing set. To address this data imbalance challenge effectively, we propose the WSSA. This approach aims to balance data distribution between training and testing sets by considering the relative weight of each website.

WSSA is designed to counter the limitations of maximum sequence length in conventional transformer-based models used for website classification. Models like BERT and RoBERTa, with a maximum sequence length of 512 tokens, often lose critical contextual information from longer web pages. Our approach involves chunking these web pages into smaller, stratified blocks and calculating the weight of each website based on the number of chunks it generates. This method ensures that larger web pages do not unduly dominate the training set, maintaining a balanced representation of each category in both training and test sets.

Our procedure begins with the entire dataset, where each web page is chunked into segments with a maximum sequence length of 500 tokens. We then calculate the weight of each website based on the number of blocks n_i it generates. Specifically, the weight w_i of a website D_i is defined as:

$$w_i = \frac{n_i}{\sum_{i=1}^{N} n_i}, \tag{4.6}$$

where n_i is the number of blocks generated from the web pages of website D_i, and $\sum_{i=1}^{N} n_i$ is the total number of blocks in the dataset. N represents the total number of websites.

To determine the frequency f_{C_ℓ} of each category C_ℓ, we sum the weights w_i of each website D_i within the same category:

$$f_{C_\ell} = \sum_{i=1}^{N} w_i \, \mathbf{1}_{\{D_i \in C_\ell\}}, \tag{4.7}$$

where $\mathbf{1}_{\{D_i \in C_\ell\}}$ is an indicator function, taking the value 1 if D_i belongs to category C_ℓ, and 0 otherwise.

We illustrate this calculation with an example in Table 4.1, which shows the frequency computation for a dataset with 9 websites and 2 categories (C_1 and C_2).

Table 4.1: Category Frequencies from Weighted Sums

Cat	D_i	n_i	w_i	f_{C_ℓ}
C_1	D_1	3	0.1304	0.4781
	D_2	2	0.0869	
	D_3	2	0.0869	
	D_4	4	0.1739	
C_2	D_5	1	0.0434	0.5219
	D_6	2	0.0869	
	D_7	1	0.0434	
	D_8	3	0.1304	
	D_9	5	0.2173	

With these weights and frequencies calculated, we initiate the weighted stratified splitting process. Starting with the whole dataset, we randomly select websites and sum their weights S until the cumulative sum reaches approximately 70% of the category frequency for the training set f_{C_ℓ}. We incorporate a Cauchy criterion ϵ for flexibility, ensuring the cumulative sum S falls within a small range of the threshold:

$$S = \sum_{i=1}^{k} w_i \quad \text{with} \quad 0.7 f_{C_\ell} - \epsilon \leq S \leq 0.7 f_{C_\ell} + \epsilon, \tag{4.8}$$

Each selected website is excluded from further selections. Once the cumulative sum S meets the Cauchy criterion around the threshold, we finalize the

drawn websites. If the cumulative sum S exceeds the threshold $0.7f_{C_\ell}$, we adjust by including or excluding the last drawn website based on the Cauchy criterion, or more randomly choose another website until the criterion is satisfied.

Finally, we recover the respective chunked blocks of the drawn websites to form the training set. Websites not selected are allocated to the test set. This process ensures a balanced distribution of data across training and testing sets, accounting for both website sizes and category frequencies, thus maintaining the representativeness of each category.

This method ensures a balanced and informative dataset, promoting fair and effective model evaluation. The Figure 4.1 presents the process of our proposed Weighted Stratified Split Approach (WSSA). Algorithm 1 presents the pseudo-code for our WSSA approach.

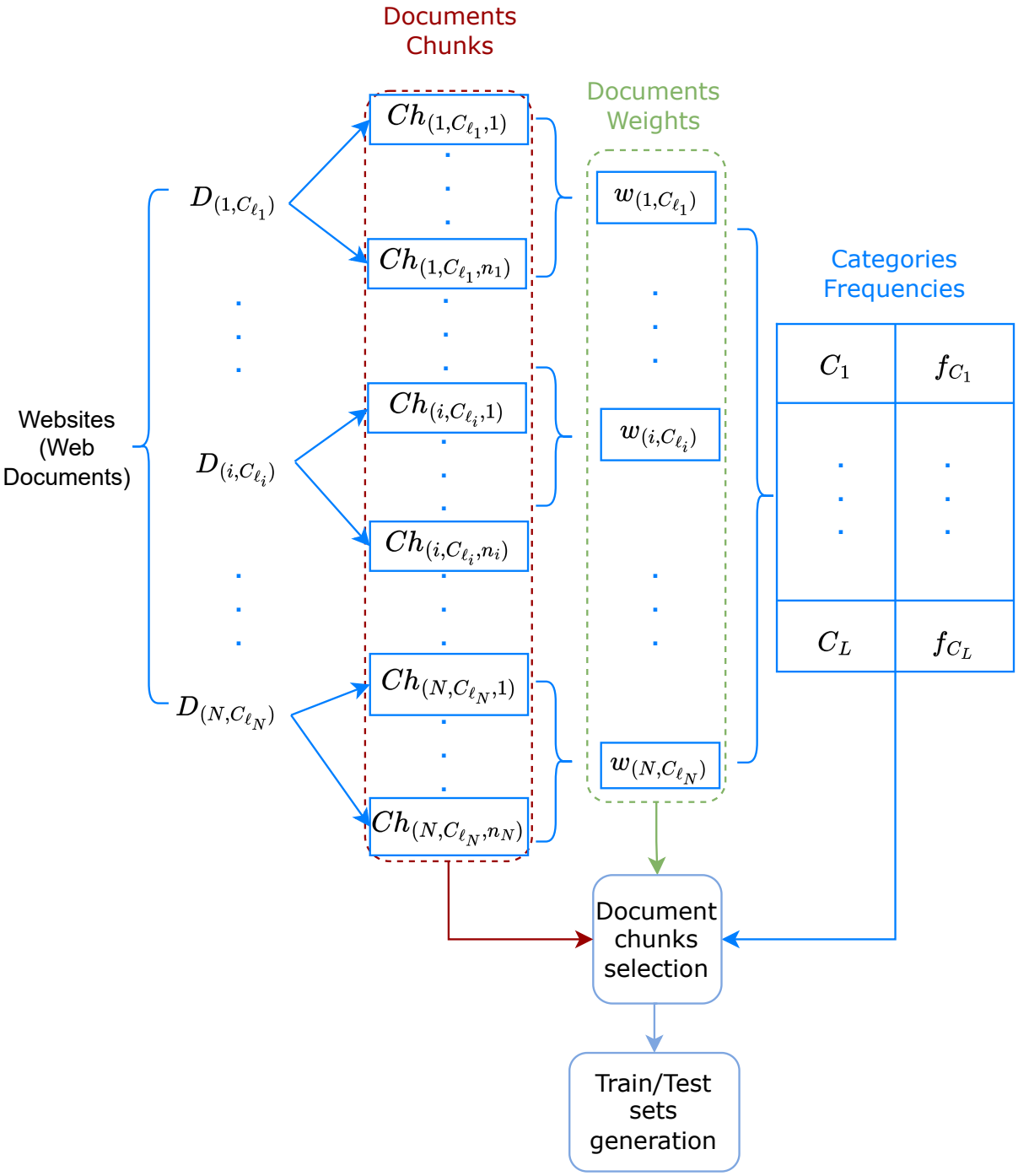


Figure 4.1: Overview of our proposed weighted stratified approach (WSSA)

Through this systematic process, training and testing sets of websites are composed to incorporate a representative cross-section of the dataset, considering both the quantity and distribution of data across categories. This approach aims to establish an equitable representation of website documents of varying sizes in the training and testing sets, effectively mitigating poten-

tial biases and fostering an accurate assessment of classification models.

Algorithm 1 Weighted Stratified Split Approach (WSSA)

Input: Dataset D of websites with their respective web pages, Threshold T for training set proportion (e.g., 0.7), Cauchy criterion ϵ for flexibility in the threshold, Maximum sequence length L for chunking (e.g., 500 tokens)

Output: Training set $Train\_Set$ and testing set $Test\_Set$

Initialize $total\_blocks \leftarrow 0$ **foreach** *website* D_i *in* D **do**
  Chunk each web page in D_i into segments of length L Calculate $n_i \leftarrow$ number of chunks generated for D_i $total\_blocks \leftarrow total\_blocks + n_i$
end
Initialize weights w_i for each website **foreach** *website* D_i *in* D **do**
  $w_i \leftarrow \frac{n_i}{total\_blocks}$
end
Initialize category frequencies f_{C_l} for each category C_l **foreach** *category* C_l *in* D **do**
  $f_{C_l} \leftarrow \sum w_i$ for all websites D_i in C_l
end
Initialize empty lists $Train\_Set \leftarrow []$ and $Test\_Set \leftarrow []$
foreach *category* C_l *in* D **do**
  $cumulative\_sum \leftarrow 0$ $selected\_websites \leftarrow []$ Shuffle the list of websites in C_l
  while $cumulative\_sum < T \times f_{C_l}$ **do**
    Randomly select a website D_i from C_l $cumulative\_sum \leftarrow cumulative\_sum + w_i$
    Append D_i to $selected\_websites$ **if** $cumulative\_sum \geq (T \times f_{C_l}) - \epsilon$ **then**
      break
    end
  end
  foreach *website* D_i *in* $selected\_websites$ **do**
    Add chunked blocks of D_i to $Train\_Set$
  end
  foreach *remaining website* D_i *in* C_l **do**
    if $D_i \notin selected\_websites$ **then**
      Add chunked blocks of D_i to $Test\_Set$
    end
  end
end
return $Train\_Set, Test\_Set$

4.3.3 Website Classification through Chunked Web Pages

In the evolution of WeCA (see chapter:3), we have adapted to a more granular method by integrating the classification of chunked web pages. This adaptation is pivotal as we move beyond page-level classification to aggregate these chunks for a comprehensive website classification. The Majority Voting strategy has been instrumental in this process, where each chunked

web page acts as a micro-classifier within its parent website, casting votes towards the overall classification.

Chunked Web Pages as Micro-Classifiers

In our refined approach, web pages are dissected into smaller chunks, aligning with the limitations of traditional transformer-based models and addressing the information-dense nature of web content. These chunks serve as micro-classifiers. Their combined classification outcomes are pivotal to understanding the broader thematic elements of a website. This nuanced strategy accommodates the varied and significant aspects of different web page sections, ensuring that even the subtlest thematic shifts within a website are accounted for in the classification process.

Adopting Majority Voting for Chunked Classification

The Majority Voting strategy is particularly well-suited for this refined approach as we have seen in Section 3.4. Each chunk casts a vote for its classification outcome, contributing to the majority consensus for the website's category. The website's final classification reflects the category receiving the majority of votes from its constituent chunks. This method effectively balances the representation of content, ensuring that both extensive and concise sections within a website are equally considered. It mitigates the risk of larger web pages overshadowing smaller but potentially more thematically relevant sections.

By utilizing the Majority Voting strategy in the context of chunked web pages, we attain a balanced and democratic classification system. This system not only respects the diversity of content within each website but also aligns with the computational constraints posed by processing large-scale web data. The result is a robust and equitable framework that stands resilient in the face of web content variability, ensuring accurate and fair classification across the web landscape.

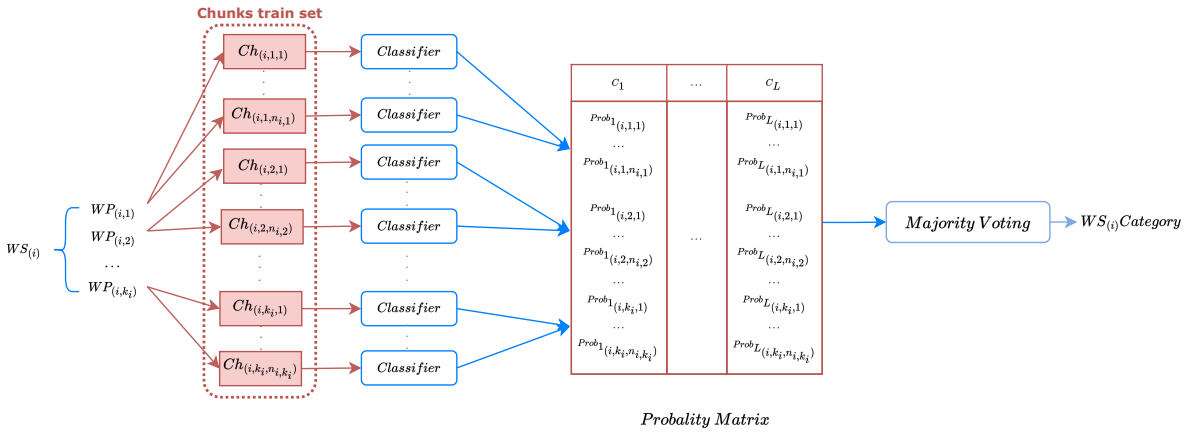


Figure 4.2: Overview of the chunks as website micro-classifiers architecture

Figure 4.2 depicts the classification pipeline for a website $WS_{(i)}$ using the chunking approach. The process begins with the i^{th} website, $WS_{(i)}$, which is decomposed into a series of web page chunks, $WP_{(i,1)}, WP_{(i,2)}, \ldots, WP_{(i,k_i)}$. Each chunk, $Ch_{(i,k,j)}, j = 1, \ldots, n_k, k = 1, \ldots, k_i$ is individually classified by a classifier. The outputs from these classifiers are then directed into a probability matrix $[Prob_{1_{(i,k,n_{i,k})}}, \ldots, Prob_{L_{(i,k,n_{i,k})}}]$, where they are aggregated using MVS. This matrix is key to the subsequent aggregation method, which synthesizes the individual chunk classifications to determine the final category for the website, $WS_{(i)}Category$. The flow of the process from chunked web pages to a unified website classification demonstrates how multiple pieces of content can be combined to yield a singular, coherent category assignment for a website.

4.4 Experiments

In our experiments, we compare the implemented approaches of automatic website classification through two main case studies. In the first case, we evaluate the performance of our classifiers over the text we gathered from the index page of each website to which we refer by depth $= 0$. In the second case study, we want to know how the classifiers would perform on the index web page of each website and also the gathered information from their

neighboring web pages in a +1 radius to which we refer to as depth = 1. In each of our experiments we used a cross validation step and a class majority voting aggregator to get the final website category since we are classifying websites and not web pages. In the following, we provide the used datasets, the evaluation metrics and our experimental results.

4.4.1 Datasets

From the Olfeo URLs dataset, we selected 10 categories while considering only English websites. The crawler we designed is made in a way to crawl the target website and also its neighboring web pages per each website. After crawling the data, we end-up with a dataset of 96741 web pages for 3368 unique websites, Table 4.2 sums-up a general view of our dataset by categories and the generated chunks of 500 tokens for each category. Figures 4.3, 4.4 and 4.5 show the data distribution for all datasets showing how the category proportions are similar to the original dataset. We follow by the text extraction from the raw HTML was performed using the same HTML tags as [290], and we additionally extracted text from the metadata tags, which often contained short texts such as the title and description of the web page, with a length ranging from 7 to nearly 100 tokens.

We compare various models for automatic website classification through two case studies. The first case evaluates the performance of classifiers based on text obtained from the index page of each website, while the second case considers both the index page (depth 0) and the information from the surrounding web pages within a radius of +1 (i.e. depth 1), these web pages are obtained from the links inside the index web page, where we only consider the first top 50 links to crawl. The objective of these two cases is to determine the effect of including additional information from surrounding web pages on the classifiers performance. Moreover, in order to compare the effectiveness of our proposed Weighted stratified split approach, we generate a baseline database and evaluate the models on each of them. For each dataset, 70% of the data is used as training data from which we take another 10% for the

validation data and finally the last 30% are used as test data. The datasets we use are described as below :

1. **Basic Split Baseline Data (BSSBD):** To generate this dataset we use the BS on the whole dataset. We split the data (i.e. web pages) following a stratified strategy in order to counter the imbalanced data bias as said in the beginning. It is noteworthy that when depth is 0 there is no need for an aggregation step, since we classify only the index web pages in this dataset and no chunking has been performed, the web pages were fed as it is.
2. **Weighted Stratified Split Approach Dataset (WSSAD):** To generate this dataset, we first take the original dataset and generate text chunks of 500 token each. We then perform the proposed WSSA on the resulted chunks of web pages to generate the train and test sets. In Table 4.2 the number of websites represents the existing unique websites referred by their index web pages which also is the number of web pages at depth 0, in the rest of the columns is stated the number of web pages respectively chunks both at depth 1.

Table 4.2: Crawled Dataset and Chunked Blocks

Category	Websites	Web Pages	$\text{Chunks}_{Depth=1}$	$\text{Chunks}_{Depth=0}$
tourism_hotel	756 (22,45 %)	20051 (20,73 %)	51300 (20,06 %)	1980 (21,20%)
business_services	587 (17,43 %)	16918 (17,49 %)	47658 (18,64 %)	1689 (18,09%)
bank_insurance	543 (16,12 %)	13623 (14,08 %)	37106 (14,51 %)	1523 (16,31%)
education	367 (10,90 %)	10327 (10,67 %)	28010 (10,95 %)	1046 (11,20%)
fashion_beauty	196 (5,82 %)	7261 (7,51 %)	17917 (7,01 %)	518 (5,55%)
cars_motor	194 (5,76 %)	6481 (6,70 %)	18572 (7,26 %)	591 (6,33%)
home_garden_interior	188 (5,58 %)	6432 (6,65 %)	15555 (6,08 %)	501 (5,36%)
escort_services	185 (5,49 %)	6204 (6,41 %)	16135 (6,31 %)	528 (5,65%)
consumer_services	202 (6,00 %)	5696 (5,89 %)	14624 (5,72 %)	557 (5,96%)
counterfeit	150 (4,45 %)	3748 (3,87 %)	8865 (3,47 %)	406 (4,35%)
Size	**3368**	**96741**	**255742**	**9339**

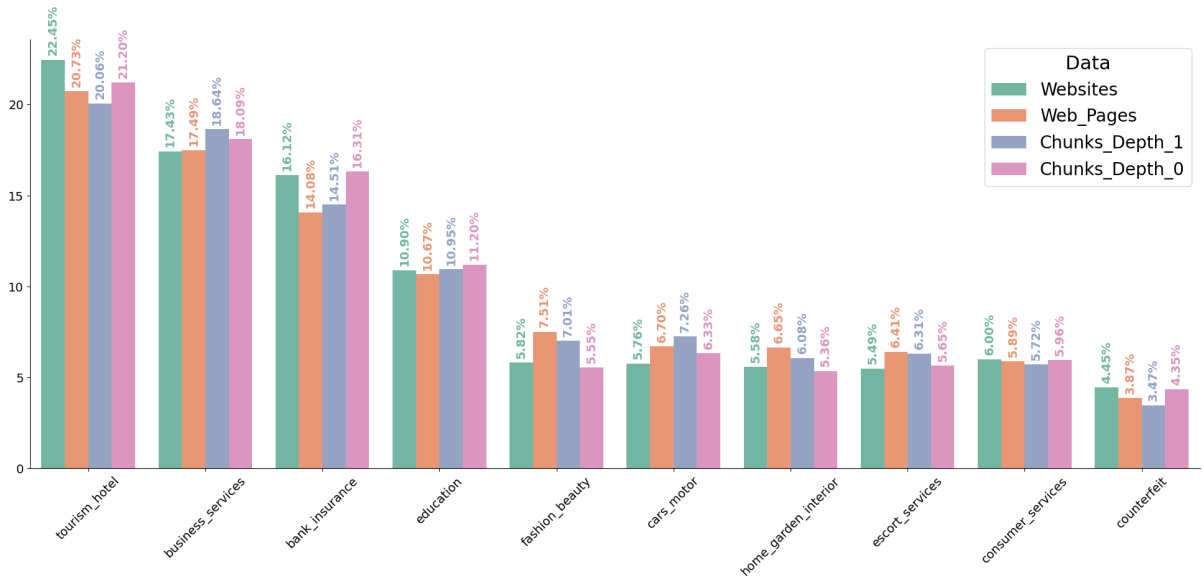
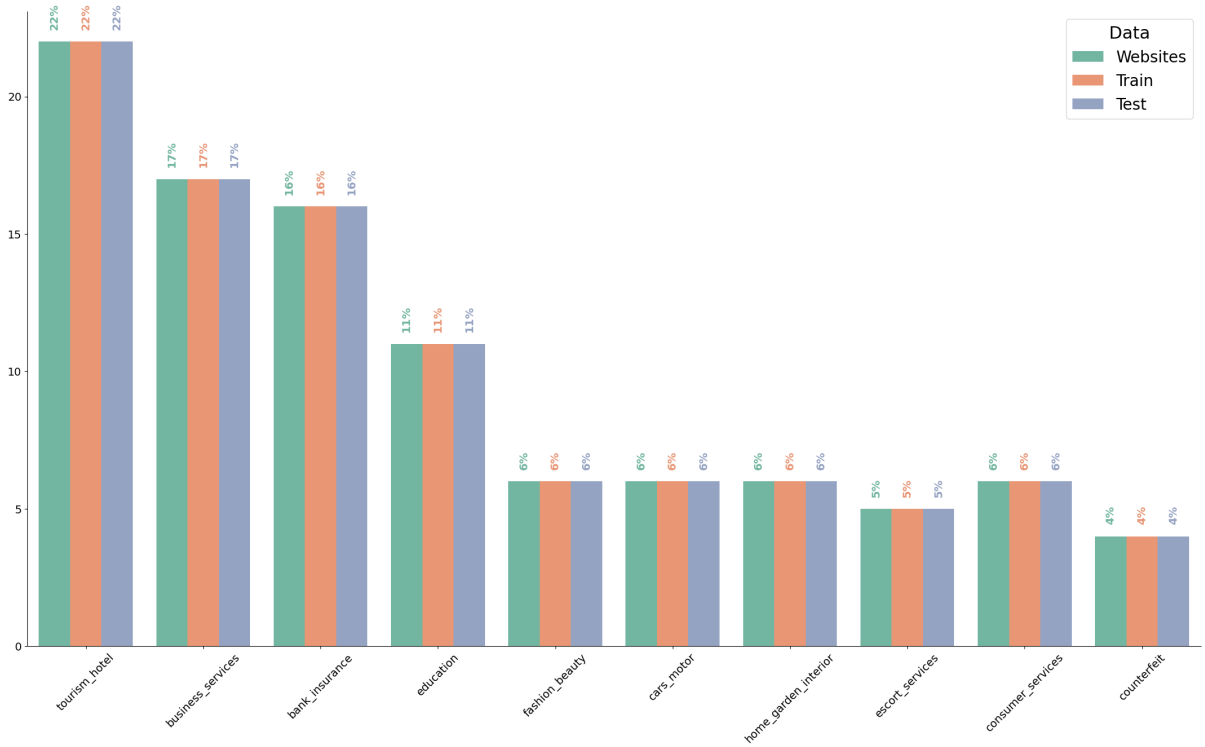


Figure 4.3: Distribution of Websites, Web Pages and Chunked blocks

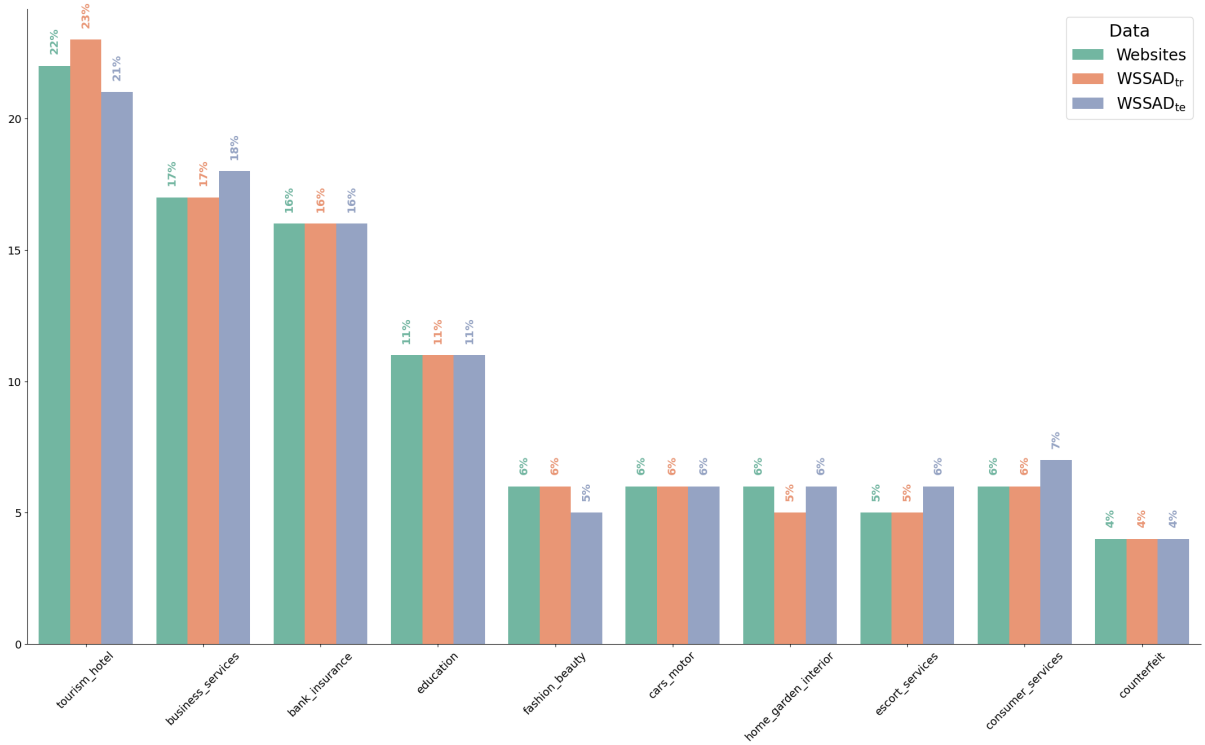
Figure 4.3 represents the distribution of the Websites, Web Pages and Chunked blocks before applying any split. Figures 4.4a and 4.5a show the distribution of websites, web pages and chunks across various categories before and after applying data splitting. In the other hand, figures 4.4a and 4.4b show the distribution of websites across BSSBD_{tr}, BSSBD_{te}, WSSAD_{tr} and WSSAD_{te}.

Figure 4.5a shows the distribution of web pages across BSSBD_{tr}, BSSBD_{te}. Figure 4.5b presents the distribution of chunks across WSSAD_{tr} and WSSAD_{te}.

Table 4.3 shows the number of websites for each category in the training and test sets, along with the overall dataset size. Additionally, Table 4.4a and Table 4.4b provide data distribution details for the BSSBD and WSSAD datasets, respectively, showing the distribution of web pages and chunked blocks in the training and test sets for each category.



(a) Distribution of Websites, BSSBD_{tr}, BSSBD_{te} sets (depth = 0)



(b) Distribution of Websites, WSSAD_{tr}, WSSAD_{te} sets (depth = 0)

Figure 4.4: Distribution of Websites before and after applying BSS and WSSA

Table 4.3: Websites for each generated train/test set after BSS and WSSA (depth = 0)

Category	$\mathbf{BSSBD}_{tr}$	$\mathbf{BSSBD}_{te}$	$\mathbf{WSSAD}_{tr}$	$\mathbf{WSSAD}_{te}$
tourism_hotel	529 (22,44%)	227 (22,45%)	543 (23,29%)	213 (20,54%)
business_services	411 (17,44%)	176 (17,41%)	396 (16,99%)	191 (18,42%)
bank_insurance	380 (16,12%)	163 (16,12%)	378 (16,22%)	165 (15,91%)
education	257 (10,90%)	110 (10,88%)	257 (11,03%)	110 (10,61%)
fashion_beauty	137 (5,81%)	59 (5,84%)	140 (6,01%)	56 (5,40%)
cars_motor	136 (5,77%)	58 (5,74%)	133 (5,71%)	61 (5,88%)
home_garden_interior	132 (5,60%)	56 (5,54%)	124 (5,32%)	64 (6,17%)
escort_services	129 (5,47%)	56 (5,54%)	126 (5,41%)	59 (5,69%)
consumer_services	141 (5,98%)	61 (6,03%)	130 (5,58%)	72 (6,94%)
counterfeit	105 (4,45%)	45 (4,45%)	104 (4,46%)	46 (4,44%)
Size	**2357**	**1011**	**2331**	**1037**

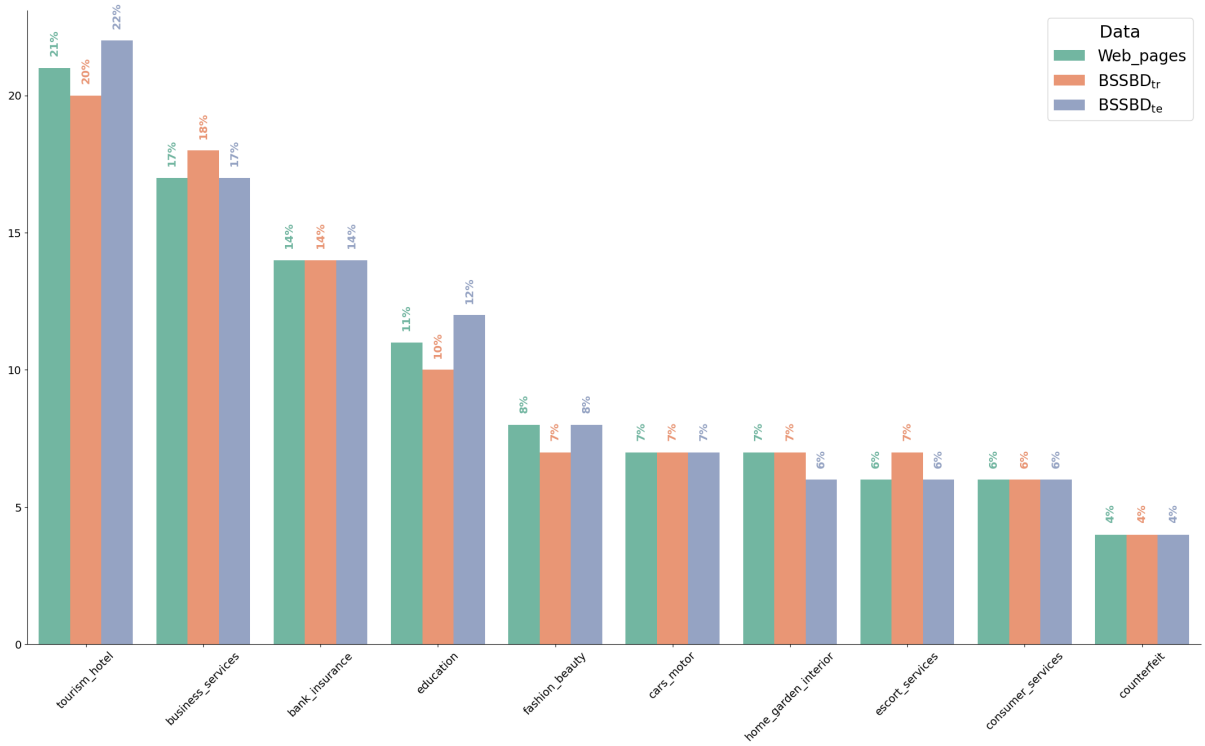
Table 4.4: Data Distribution for BSSBD and WSSAD

(a)
Web Pages Distribution for train/test sets in BSSBD (depth = 1)

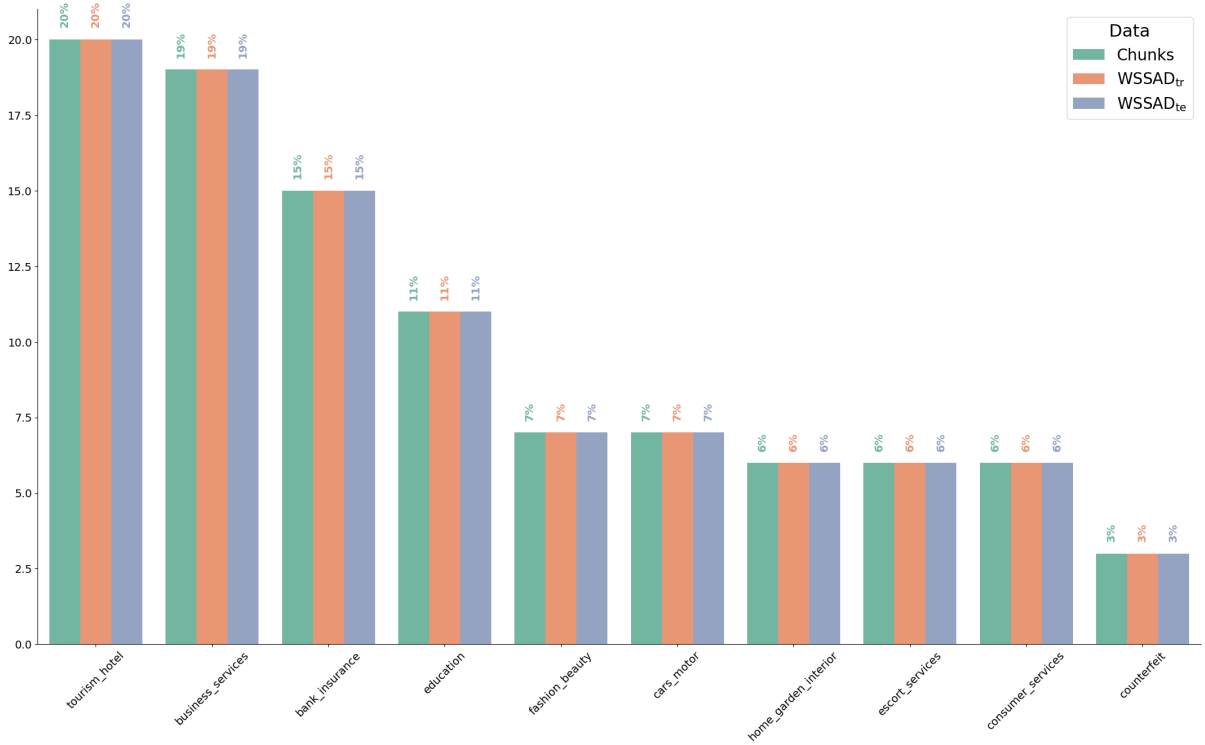
Category	$\mathbf{BSSBD}_{tr}$	$\mathbf{BSSBD}_{te}$
tourism_hotel	13944 (20,40 %)	6107 (21,51 %)
business_services	12082 (17,68 %)	4836 (17,03 %)
bank_insurance	9718 (14,22 %)	3905 (13,75 %)
education	7009 (10,25 %)	3318 (11,69 %)
fashion_beauty	5107 (7,47 %)	2154 (7,59 %)
cars_motor	4505 (6,59 %)	1976 (6,96 %)
home_garden_interior	4842 (7,08 %)	1590 (5,60 %)
escort_services	4558 (6,67 %)	1646 (5,80 %)
consumer_services	3929 (5,75 %)	1767 (6,22 %)
counterfeit	2657 (3,89 %)	1091 (3,84 %)
Size	**68351**	**28390**

(b)
Chunked blocks Distribution for train/test sets in WSSAD (depth = 1)

Category	$\mathbf{WSSAD}_{tr}$	$\mathbf{WSSAD}_{te}$
tourism_hotel	35910 (20,06 %)	15390 (20,06 %)
business_services	33360 (18,64 %)	14298 (18,63 %)
bank_insurance	25974 (14,51 %)	11132 (14,51 %)
education	19606 (10,95 %)	8404 (10,95 %)



(a) Distribution of Web Pages, BSSBD$_{tr}$, BSSBD$_{te}$ sets (depth = 1)



(b) Distribution of Chunks, WSSAD$_{tr}$, WSSAD$_{te}$ sets (depth = 1)

Figure 4.5: Distribution of Chunks and Web Pages before and after applying BSS and WSSA

4.4.2 Evaluation Metrics

In this section, we present the main metrics we use to evaluate our models. After establishing the confusion matrix for each experiment. We then measure the accuracy. We then follow with a comparison inter-classifiers over the same datasets with the Cochran's Q test in order to establish that the obtained results are consistently significant.

Cochran's Q test

Cochran's Q test can be regarded as a generalized version of McNemar's test that can be applied to evaluate multiple classifiers. In a sense, Cochran's Q test is analogous to ANOVA for binary outcomes.

To compare more than two classifiers, we can use Cochran's Q test, which has a test statistic Q that is approximately, (similar to McNemar's test), distributed as χ^2 with $L-1$ degrees of freedom, where L is the number of models we evaluate (since $L=2$ for McNemar's test, McNemar's test statistic approximates a χ^2 distribution with one degree of freedom). More formally, Cochran's Q test tests the null hypothesis that there is no difference between the classification accuracies (i.e., $H_0 : p_1 = p_2 = \cdots = p_L$).

Let $\{D_1,\ldots,D_L\}$ be a set of classifiers who have all been tested on the same dataset. If the L classifiers don't perform differently, then the following Q statistic is distributed approximately as chi-square with $L-1$ degrees of freedom :

$$Q_C = (L-1)\frac{L\sum_{i=1}^{L} G_i^2 - T^2}{LT - \sum_{j=1}^{N_{ts}}(L_j^2)}. \tag{4.9}$$

Here, N_{ts} is the size of the test dataset, G_i is the number of objects out of N_{ts} correctly classified by $D_i, i = 1,\ldots L$; L_j is the number of classifiers out of L that correctly classified object $z_j \in Z_{ts}$, where $Z_{ts} = \{z_1, ...z_{N_{ts}}\}$ is the test dataset on which the classifiers are tested on; and T is the total number of correct number of votes among the L classifiers [224, 328] :

$$T = \sum_{i=1}^{L} G_i = \sum_{j=1}^{N_{ts}} L_j. \tag{4.10}$$

To perform Cochran's Q test 4.9 4.10, we typically organize the classifiers predictions in a binary $N_{ts} \times L$ matrix M. The ijth entry of such matrix $M_{i,j}$ is 0 if a classifier D_j has misclassified a data example (vector) z_i and 1 otherwise (if the classifier predicted the class label $l(z_i)$ correctly) [328]. Hence,

$$G_i = \sum_{i=1}^{N_{ts}} M_{i,j}, j = 1, \ldots, L. \tag{4.11}$$

And,

$$L_i = \sum_{j=1}^{L} M_{i,j}, i = 1, \ldots, N_{ts}. \tag{4.12}$$

4.4.3 Classification Results

All the experiments are implemented using Python and run through the super calculator ROMEO[1]. Transformer models were fine-tuned using huggingface library (version 4.29.2). Due to resources limitations, we used a max sequence length for Longformer and BigBird of 1024 with a training and validation batch size equal to 4 (low GPU memory) and the learning rate was 1e-5. For BERT and RoBERTa the max sequence length is by default 512, the training batch size was 16 and the validation batch size was 32 and the learning rate for both models was 5e-5. For the rest of the parameters, we kept the default ones. All the models were fine-tuned for 40 epochs.

The results in Table 4.6 present the accuracies obtained after applying the Majority Voting Strategy (MVS) to aggregate the predictions of classification models. The focus here is on the performance of the models on the Weighted stratified split approach (WSSAD), as compared to the Basic Split Baseline Dataset (BSSBD).

[1]https://romeo.univ-reims.fr/

Table 4.5: Model Fine Tuning Time (with FT for Fine Tuning and IT for Inference Time)

		BSSBD		WSSAD	
Depth	**Clf**	**FT (mn)**	**IT (s)**	**FT (mn)**	**Inf (s)**
0	**BERT**	15,8	0,0033	14,9	0,0012
	RoBERTa	4,9	0,0032	13,1	0,0012
	Longformer	26,7	0,0208	126	0,0203
	BigBird	54,6	0,0210	498	0,0891
1	**BERT**	48,7	0,0033	216	0,0011
	RoBERTa	48,3	0,0031	216	0,0012
	Longformer	204	0,0206	2880	0,0203
	BigBird	288	0,0207	8640	0,0908

Across different depths and classifiers, the accuracies achieved on the WSSAD dataset consistently outperform those obtained on the BSSBD dataset. Notably, the accuracies on WSSAD are higher for most combinations of depth and classifier. For instance, considering the results at depth 0, both BERT and RoBERTa models achieve higher accuracies on WSSAD (93.09% and 93.10%, respectively) compared to BSSBD (89.55% and 88.33%, respectively). This improvement is also observed for the Longformer and BigBird models, where the accuracies on WSSAD (92.96% and 91.59%, respectively) surpass those on BSSBD (89.15% and 82.77%, respectively).

Moving to depth 1, we see a similar trend. BERT and RoBERTa models attain higher accuracies on WSSAD (91.90% and 92.14%, respectively) compared to BSSBD (90.70% and 89.96%, respectively). While the Longformer model exhibits a slightly lower accuracy on BSSBD (87.60%) compared to WSSAD (90.71%), the BigBird model demonstrates a noteworthy increase in accuracy on WSSAD (86.59%) compared to BSSBD (85.42%).

These results highlight the effectiveness of the weighted stratified split approach employed in the WSSAD dataset. By considering the content and context of web page chunks, the WSSAD dataset provides a more comprehensive representation of web pages, enabling the classification models to make

Table 4.6: Accuracies after MVS

Depth	Clf	BSSBD	WSSAD
0	**BERT**	**89,55%**	93,09%
	RoBERTa	88,33%	**93,10%**
	Longformer	89,15%	92,96%
	BigBird	82,77%	91,59%
1	**BERT**	**90,70%**	91,90%
	RoBERTa	89,96%	**92,14%**
	Longformer	87,60%	90,71%
	BigBird	85,42%	86,59%

more accurate predictions. The consistently higher accuracies obtained on the WSSAD dataset demonstrate its value in improving the performance of classification models for web page classification tasks.

Therefore, the weighted stratified based split approach in the WSSAD dataset showcases its significance in enhancing the accuracy of classification models.

4.4.4 Cochran's Q Test Results

In our study, we formulate the following hypotheses for the Cochran's Q test:

- **Null Hypothesis (H0) :** There are no significant differences in the performances of the classifiers on the classification task. In other words, the classifiers predictions are consistent across instances, and any observed differences are due to random fluctuations.
- **Alternative Hypothesis (H1) :** There are significant differences in the performances of the classifiers on the classification task. This suggests that certain classifiers demonstrate superior or inferior performance compared to others, and the observed differences are not solely attributable to chance.

By subjecting our data to the Cochran's Q test, we aim to rigorously assess the significance of these differences and gain a deeper understanding of the

relative strengths and weaknesses of the classifiers under consideration. This statistical analysis will enable us to make informed decisions about the most effective classifiers for our task.

The table presented below displays the outcomes of the Cochran's Q tests conducted for all classifiers across each dataset. Our initial focus is on comparing the classifiers across different datasets at varying depths. To ensure the applicability of the Cochran's Q test for classifiers trained on different datasets, we create two new test datasets. Subsequently, we evaluate the fine-tuned BERT and RoBERTa models on these newly generated test datasets. This approach allows us to comprehensively assess the performances of the classifiers, facilitating fair and meaningful comparisons even when dealing with varying training datasets.

Table 4.7: Cochran's Q Test Results

(a)
Cochran's Q Test Inter-Datasets

Depth	Test dataset	Cochran's Q	p-value
0	**BSSBD**	45.887	5.99e-10
	WSSAD	50.464	6.36e-10
1	**BSSBD**	3.674	0.2988
	WSSAD	131.372	2.97e-15

(b)
Cochran's Q Test for BERT and RoBERTa

Depth	Test dataset	Cochran's Q	p-value
0	**BSSBD**$_{new}$	87.38	3.22e-12
	WSSAD$_{new}$	90.55	7.47e-12
1	**BSSBD**$_{new}$	85.40	6.06e-13
	WSSAD$_{new}$	140.592	5.31e-21

The results of the Cochran's Q test, as presented in Table 4.7, provide valuable insights into the performance of classifiers across different datasets and depths.

The Cochran's Q test results provide a clear demonstration of the significant

impact of the WSSAD on classifier performance evaluation. For the inter-dataset (Table 4.7a), at Depth 0, the BSSBD dataset yielded a Cochran's Q value of 45.887 with a p-value of 5.99e-10, indicating significant differences in classifier performances. Similarly, the WSSAD dataset showed a Cochran's Q value of 50.464 with a p-value of 6.36e-10, reinforcing the rejection of the null hypothesis (H0) and confirming substantial differences in performance. At Depth 1, however, the BSSBD dataset produced a Cochran's Q value of 3.674 with a p-value of 0.2988, suggesting no significant differences in classifier performances. In contrast, the WSSAD dataset at the same depth exhibited a Cochran's Q value of 131.372 with a p-value of 2.97e-15, highlighting significant differences.

The intra-dataset (Table 4.7b) analysis further supports these findings. For Depth 0, the BSSBD_{new} dataset resulted in a Cochran's Q value of 87.38 with a p-value of 3.22e-12, while the WSSAD_{new} dataset showed a Cochran's Q value of 90.55 with a p-value of 7.47e-12, both indicating significant performance differences. At Depth 1, the BSSBD_{new} dataset had a Cochran's Q value of 85.40 with a p-value of 6.06e-13, and the WSSAD_{new} dataset presented a Cochran's Q value of 140.592 with a p-value of 5.31e-21, again demonstrating significant differences.

These results underscore the effectiveness of the WSSAD approach in revealing performance variations that are not detected by the BSSBD dataset. The significant differences highlighted by the WSSAD method at both depths emphasize the importance of employing appropriate stratified data splitting techniques for achieving more reliable and insightful evaluations of classifier performance.

Overall, the Cochran's Q test illustrates the critical role of dataset selection and model choice in evaluating classifier performance. It demonstrates that different datasets and models can result in significant variations in classifier performances. Therefore, researchers and practitioners should exercise caution in their selection of datasets and models to ensure accurate and reliable evaluations in natural language processing tasks.

4.4.5 RoBERTa-based WeCA Performance on a New Test Dataset

In the following we focus more in depth in RoBERTa's performance to asses both the baseline BSSBD fine-tuned model and WSSAD based fine-tuned model. To do so, we generate a New Test Dataset (NTD) as described in Table 4.8. We then use $\text{RoBERTA}_{\text{BSSBD}}$ and $\text{RoBERTA}_{\text{WSSAD}}$ in WeCA (using MVS) and apply them to website classification (Depth 0 and Depth 1). In the first experiment, we use the web pages of a website as input, while in the second, we use the chunks of a website. The obtained results are summarized in Table 4.9 which presents the accuracy, Cochran's Q test statistics, and corresponding p-values for RoBERTa$BSSBD$ and RoBERTa$WSSAD$ at two different depths. We follow the same Hypothesis as seen in Subsection 4.4.4.

Table 4.8: New Test Dataset (NTD)

Category	Websites	Web Pages	Chunks	$\text{Chunks}_{depth=0}$
tourism_hotel	37 (9.64%)	311 (10.05%)	3841 (11.40%)	331 (9,68%)
business_services	42 (10.94%)	256 (8.28%)	2110 (6.26%)	239 (6,99%)
bank_insurance	48 (12.50%)	399 (12.90%)	4709 (13.98%)	518 (15,15%)
education	33 (8.59%)	365 (11.80%)	4947 (14.69%)	389 (11,37%)
fashion_beauty	39 (10.16%)	302 (9.76%)	3568 (10.59%)	350 (10,23%)
cars_motor	31 (8.07%)	280 (9.05%)	3333 (9.89%)	313 (9,15%)
home_garden_interior	45 (11.72%)	411 (13.29%)	5233 (15.55%)	575 (16,81%)
escort_services	30 (7.81%)	230 (7.44%)	2735 (8.12%)	217 (6,35%)
consumer_services	36 (9.38%)	289 (9.34%)	3421 (10.16%)	322 (9,42%)
counterfeit	43 (11.20%)	200 (6.47%)	1094 (3.25%)	166 (4,85%)
Sum	**384**	**3093**	**33691**	**3420**

Evaluation on Web Pages

The performance of RoBERTa models on web pages is shown in Table 4.9a. At depth 0, RoBERTa_{WSSAD} achieved an accuracy of 91.10%, compared to 87.81% for RoBERTa_{BSSBD}. The Cochran's Q test statistic for this depth was 15.551 with a p-value of 3.89e-11, indicating a statistically significant

Table 4.9: RoBERTa's Performance on NTD with Cochran's Q Test

(a) Evaluation of Websites Classification using Web Pages as Input

Depth	Model	Accuracy	Cochran's Q	p-value
0	**RoBERTa**$_{BSSBD}$	**87,81 %**	15.551	**3.89e-11**
	RoBERTa$_{WSSAD}$	**91,10 %**		
1	**RoBERTa**$_{BSSBD}$	**89,76 %**	67.137	**6.40e-10**
	RoBERTa$_{WSSAD}$	**92,12 %**		

(b) Evaluation of Websites Classification using Chunked Blocks as Input

Depth	Model	Accuracy	Cochran's Q	p-value
0	**RoBERTa**$_{BSSBD}$	**89,77 %**	30.112	**2.09e-11**
	RoBERTa$_{WSSAD}$	**92,54 %**		
1	**RoBERTa**$_{BSSBD}$	**91,09 %**	127.92	**4.66e-9**
	RoBERTa$_{WSSAD}$	**94,01 %**		

improvement. Therefore, we reject the null hypothesis (H0) and accept the alternative hypothesis (H1), as there are significant differences in the performances of the classifiers.

At depth 1, RoBERTa$_{WSSAD}$ achieved an accuracy of 92.12%, whereas RoBERTa$_{BSSBD}$ achieved 89.76%. The Cochran's Q test statistic was 67.137 with a p-value of 6.40e-10, further demonstrating a significant improvement. Again, we reject the null hypothesis (H0) and accept the alternative hypothesis (H1).

Evaluation on Chunked Blocks

The performance of RoBERTa models on chunked blocks is shown in Table 4.9b. At depth 0, RoBERTa$_{WSSAD}$ achieved an accuracy of 92.54%, compared to 89.77% for RoBERTa$_{BSSBD}$. The Cochran's Q test statistic for this depth was 30.112 with a p-value of 2.09e-11, indicating a significant improvement. Hence, we reject the null hypothesis (H0) and accept the alternative hypothesis (H1).

At depth 1, RoBERTa$_{WSSAD}$ achieved an accuracy of 94.01%, whereas RoBERTa$_{BSSBD}$ achieved 91.09%. The Cochran's Q test statistic was 127.92 with a p-value of 4.66e-9, again indicating a significant improvement. We therefore reject

the null hypothesis (H0) and accept the alternative hypothesis (H1).

In Summary, the results clearly show that RoBERTa, when fine-tuned using WSSAD, outperforms the BSSBD on both web pages and chunked blocks. The Cochran's Q test results further validate the statistical significance of the performance improvements observed with WSSAD. These findings highlight the effectiveness of the WSSA in improving the robustness and accuracy of RoBERTa for website classification tasks. By rejecting the null hypothesis (H0) in favor of the alternative hypothesis (H1), we confirm that the observed improvements are not due to random chance but reflect genuine enhancements in model performance.

4.5 Conclusion

One of the most known open issue in the research community is processing large-scale text sequences in many NLP tasks. In this chapter, we proposed a novel simple statistical approach, namely Weighted Stratified Split Approach (WSSA), for website classification. We evaluated its performance using popular language models, including BERT, RoBERTa, Longformer and BigBird, on the WSSAD dataset. We compared the results to the baseline dataset (BSSBD) and observed that BERT and RoBERTa achieved high accuracies. However, their maximum sequence length limitation hampered their performance on longer web pages.

To address this limitation, we introduced the WSSA split, which divides web pages into stratified chunks, allowing for the retention of contextual information while maintaining a balanced distribution of categories. WSSA approach outperforms Longformer and BigBird models, which struggle with longer texts and complex web page content.

WSSA achieves high accuracies with Longformer and RoBERTa for depth-0 classification tasks and surpasses BigBird for depth-1 classification tasks. However, we acknowledge the slight discontinuities introduced by the chunk-

ing process and the dependency of the majority voting strategy on individual model accuracies and noisy chunks. We also notice that the best results are obtained through the combination of index web pages, RoBERTa model and our WSSAD dataset. This result shows how efficient website classification can be done while being low-cost in resources and time consumption unlike other models.

Overall, our study demonstrates the effectiveness of the WSSA split in improving website classification accuracy and overcoming the limitations of BERT and RoBERTa models. These findings have implications for applications such as information retrieval and content filtering.

Chapter 5

Application of WeCA to the Large-Scale Olfeo Database

5.1 Introduction

Large-scale data presents a unique set of challenges that can complicate data processing, analysis, and storage. As datasets grow in volume, variety, and velocity, traditional data processing techniques often fall short, requiring innovations in both technological approaches and analytical methods. The challenges of managing large-scale data include ensuring data quality and integrity, dealing with heterogeneous data types, and extracting meaningful insights in a timely manner. Managing large-scale data presents a set of challenges that can complicate data processing, analysis, and storage.

In the context of ML and NLP, these challenges are magnified. Developing models that can effectively learn from and make predictions on large-scale datasets requires not only sophisticated algorithms but also strategic data management to ensure that the models are both accurate and robust. As datasets grow, maintaining the balance between computational resource allocation and model performance becomes increasingly critical. Addressing these challenges is essential for advancing the capabilities of ML systems and harnessing the full potential of large-scale data.

In this chapter, we explore the application of our developed techniques for website classification, which sets the stage for a thorough analysis and practi-

cal implementation on datasets of different sizes. We start by applying these techniques to a larger dataset as an initial step. This initial application serves as a foundation, enabling us to assess the effectiveness and efficiency of our methods on a broader scale.

This chapter is organized as follows: in Section 1, we present an overview of the website classification system. Section 2 describes the full dataset and the themed datasets along with their respective categories. In Section 3, we present our experiments and results. Finally, in Section 4, we conclude with our findings and perspectives.

5.2 Overview of the website classification system

The proposed system architecture (see Figure 5.1) is designed to efficiently handle large-scale data processing and machine learning operations. It consists of three main components: Data Orchestration, Model Orchestration, and Machine Learning Operations (MLOps). Data Orchestration manages the acquisition, processing, and storage of data, ensuring a seamless flow of data throughout the system. Model Orchestration oversees the training and evaluation of machine learning models, focusing on optimizing their performance and adaptability by varying the data used to answer the task of website classification. MLOps, or Machine Learning Operations, is a set of practices that aims to deploy and maintain machine learning models in production reliably and efficiently.This step is essential for the deployment and monitoring of these models, incorporating continuous validation, deployment strategies, and performance analysis to maintain and enhance the system's efficacy. In the following, we introduce each component in details.

5.2.1 Data Orchestration (DOr)

Data orchestration is crucial for managing and facilitating the flow of data within proposed system It encompasses a series of sophisticated operations

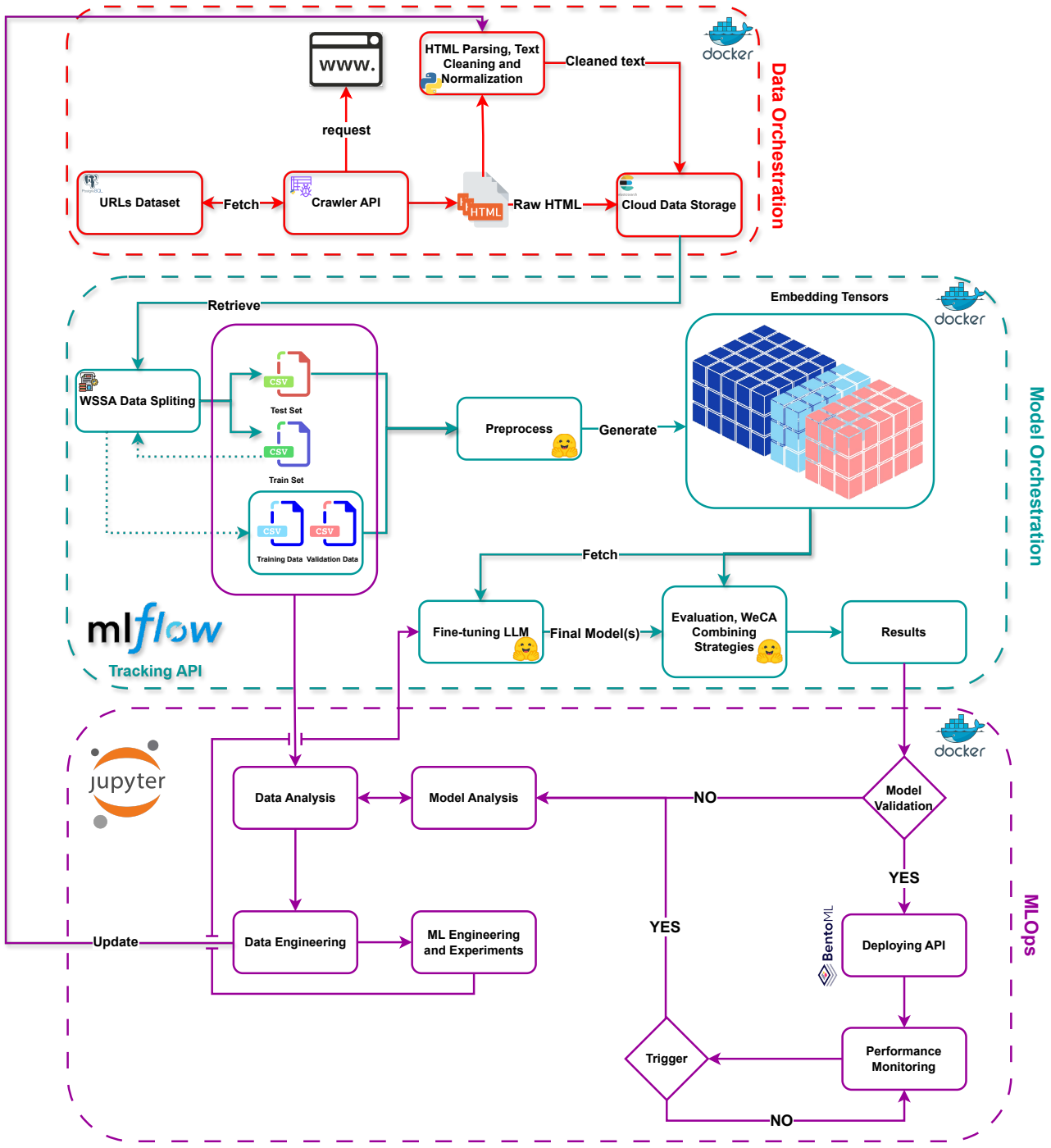


Figure 5.1: Overview of the implemented system

that begin with data acquisition from a distributed source and extend to the preparation of this data for further processing stages. This subsection focuses on the initial stages of data orchestration, specifically detailing our crawler's functionality and its integration into the data management workflow.

Crawler API

The Crawler API is a crucial part of our data orchestration process, designed to efficiently gather and process raw HTML content from specified web sources. Implemented internally in Python using the Scrapy framework, the API utilizes asynchronous processing for concurrent downloads, significantly speeding up data collection. The system follows a multi-tier architecture, separating data collection, processing, and storage layers. Ansible is used for configuration management and deployment automation, ensuring consistent environments across development, staging, and production.

The crawler targets pre-defined URLs and expands its search to include up to 50 neighboring pages within the same domain. Multi-threading allows it to handle multiple web pages simultaneously, reducing the total time required for data collection. The key tools used include Scrapy for crawling, BeautifulSoup for HTML parsing, NLTK for text processing, Dask for data manipulation, and Elasticsearch for scalable data storage and real-time analysis. Docker is used to containerize the application, ensuring consistency across different environments and simplifying deployment.

Error handling includes a retry mechanism for transient network issues and an alert system to notify administrators of persistent problems. The collected raw HTML data is stored in Elasticsearch, facilitating easy access, retrieval, and real-time data analysis. Preprocessing involves HTML parsing, text extraction, and normalization using BeautifulSoup and NLTK, transforming the data into a structured format suitable for model training and analysis.

Performance metrics indicate the crawler processes approximately 200 pages per minute, collecting around 1GB of data per hour. Docker ensures efficient resource utilization by isolating crawling tasks in containers, which can be scaled horizontally as needed. Automated deployment with Ansible ensures consistent and repeatable setups, with the architecture allowing for quick deployment of additional crawler instances to handle increased load or distribute tasks across multiple servers. This setup provides the foundation of the data collection and processing tasks.

Data Cleaning

In the task of website classification, preprocessing HTML content is essential to optimize the performance and accuracy of large language models (LLMs). In the following we present key steps of the preprocessing task necessary for effective model training.

The preprocessing begins with data cleaning to ensure the quality and integrity of the training data to ensure the elimination of noisy and redundant data, inconsistencies, and irrelevant elements:

- **Removing Duplicate Entries:** Eliminates redundant data that could bias the model.
- **Handling Missing or Erroneous Values :** Incorrect data or gaps in HTML can lead to inaccurate model training. Techniques include imputation (filling missing values based on other data points) and deletion (removing sections of data that lack essential information).
- **Parsing :** Involves analyzing the HTML structure to extract meaningful text. It transforms HTML elements into structured data usable for classification, extracting valuable information from elements like headers, paragraphs, and links.
- **Normalization :** May involve converting text to a common case, though the decision to use lowercase or maintain original casing depends on the specifics of the LLM. This step standardizes textual data to ensure uniformity and consistency, minimizing complexity for models. It also standardizes other elements like numerical data and dates to create a coherent dataset.
- **Language Detection :** Detecting the language of the website's content is crucial before further processing. This step tailors subsequent text processing to the specific linguistic rules and nuances of the content's language, important for multilingual pages.

These preprocessing steps are essential for efficiently fine-tuning the LLMs for

website classification. By implementing processes such as cleaning, parsing, normalizing, and detecting the language of the data, these actions prepare the models to accurately learn from and generalize the training data.

Data Storage

We store the raw HTML web pages in Elasticsearch (ES) cloud data storage to facilitate analysis when needed and to accommodate the dynamic evolution of the web. This approach ensures the availability of the data, eliminating the need to repeatedly crawl the same data. Additionally, the cleaned text data extracted from the HTML is also stored in the same cloud storage, providing organized access to both raw and processed data for efficient retrieval and use in various applications. This method streamlines the workflow and enhances the effectiveness of data management practices.

5.2.2 Model Orchestration (MO)

Model Orchestration within our system architecture is primarily focused on managing the training and evaluation of machine learning models efficiently. This component is crucial for ensuring that the models we develop are both effective and reliable, capable of performing well across various datasets and operational scenarios. Through MO, we handle each model's lifecycle, from its initial configuration and training through to its thorough evaluation against performance metrics.

The MO is designed to streamline the deployment of models and their integration with the overall system, including the Data Orchestration and MLOps components. It supports consistent model performance and aids in the seamless transition of models from the development stage to production. This structured approach ensures that all models meet the necessary standards before they are deployed, contributing significantly to the robustness and reliability of our machine learning operations.

WSSA Data Splitting

In this step, the process begins with the retrieval of cleaned data from the cloud database, ensuring that the data used for model training and evaluation is of high quality and free from inconsistencies Once retrieved, this data undergoes a crucial step od data splitting using the Weighted Stratified Split Approach (WSSA), a method we have developed and detailed in Chapter 4. This approach is designed to split the data in a way that maintains a representative distribution of different classes or categories, which is essential for training unbiased and generalizable models.

Preprocess

Within the preprocessing block, we employ the *HuggingFace* framework to process the data comprehensively from start to finish, culminating in the generation of embeddings that are crucial for training our LLMs. The *HuggingFace* framework is instrumental in transforming raw data into a format that can be effectively utilized by the classification mechanism. This preprocessing involves several key steps outlined by the *HuggingFace* framework (tokenizing text into consistent tokens, converting them to numerical representations, assembling these into tensors, processing in uniform batches, adding special tokens for model requirements, and generating attention masks and token type IDs for relevant data identification).

This thorough and methodical approach ensures that the data is not only prepared for model ingestion but is also optimized for the best performance of the classification mechanism. The use of the *HuggingFace* framework standardizes this process, making it scalable and repeatable across different datasets and model architectures, ultimately enhancing the robustness and reliability of our predictive models.

Fine-tuning and Evaluation

In the fine-tuning block, the primary focus is on adapting a pre-trained large language model (LLM) to our specific data and task requirements using the *HuggingFace* framework. This fine-tuning is crucial because it allows the general capabilities of the pre-trained model to be specialized to the nuances

and specifics of our dataset, which enhances its applicability and performance in our targeted scenarios.

- **Model Preparation** : We begin by selecting a suitable pre-trained model from the *HuggingFace* model hub, which has been initially trained on a large and diverse corpus. This pre-training provides the model with a broad understanding of language, which we then tailor to our specific needs.
- **Hyperparameter Optimization** : Before fine-tuning begins, we determine the optimal hyperparameters that will govern the fine-tuning process. These parameters, such as learning rate, batch size, and number of epochs, are crucial as they significantly affect the model's learning dynamics and the effectiveness of the fine-tuning.
- **Fine-Tuning Process** : Using our preprocessed and split data, the fine-tuning process involves making targeted adjustments to the model's weights. Unlike initial training, fine-tuning requires fewer epochs and adjustments because the model is already knowledgeable about the language features. The goal here is to refine this knowledge so the model can effectively apply it to the specific characteristics of our data.

By specifically adjusting how the model processes and responds to our data, fine-tuning enhances the model's ability to generalize well on new, unseen data while maintaining high accuracy on the type of content it will encounter in deployment.

Using the *HuggingFace* framework throughout this process not only facilitates these steps but also ensures they are conducted efficiently and with the support of a robust, community-driven toolkit.

Finally, in the evaluation block, WeCA is utilized to determine the final classification of websites which is detailed in Chapter 3.

Tracking API

The entire process including data splitting, model training, fine-tuning, and evaluation is encapsulated and managed by the *MLflow Tracking API*. This

API provides a systematic way to log and store all relevant data and model metrics throughout the model's lifecycle. MLflow contributes to the orchestration and subsequent stages by storing model checkpoints, dataset indices, classification results, and parameters to ensure reproducibility, performance tracking, and integration with MLOps for efficient deployment and continuous improvement (for a detailed review please refer to MLflow[1]).

By leveraging *MLflow Tracking API*, our Model Orchestration component not only enhances the management and traceability of our machine learning models but also supports the broader goal of maintaining a robust, scalable, and transparent AI system. This integration plays a pivotal role in streamlining the entire lifecycle of our models, from development to deployment.

5.2.3 Machine Learning Operations (MLOps)

In the MLOps component, model validation serves as a pivotal decision point that dictates subsequent actions based on the analysis of classification results from the MO component. This component is structured to handle both scenarios where the model is validated as either sufficient or insufficient according to predefined criteria (i.e. model's accuracy through time). Here's how the process unfolds:

Model Validation and Conditional Analysis upon receiving classification results, initial validation checks if the model meets performance criteria (i.e. if the model's accuracy is above a certain acceptable threshold, 87% in this proposed system). If the model fails, detailed data and model analysis are conducted to identify and resolve issues. If the model passes, it progresses to deployment using BentoML API (for more details please refer to BentoML[2]).

Deployment and Performance Monitoring after deployment, continuous performance monitoring detects deviations from expected standards. If performance triggers are activated, the system initiates analysis to ad-

[1]https://mlflow.org/docs/latest/introduction/index.html
[2]https://docs.bentoml.com/en/latest/

dress potential issues, ensuring the model operates effectively and adapts to evolving conditions.

This structured approach in the MLOps component ensures that our models are not only developed and deployed efficiently but are also maintained proactively to adapt to evolving conditions and requirements. This methodology supports continuous improvement and alignment with operational goals, ensuring the long-term efficiency and reliability of the classification mechanisms.

5.3 Dataset

The Olfeo[3] dataset curated and analyzed in this chapter encapsulates a comprehensive web crawl conducted from 2015 to 2022. This dataset was constructed with the objective of exploring web dynamics and potential biases, all while adhering to rigorous data quality protocols. The crawl was executed to a depth of one link from the starting pages using a custom-designed crawler API, which is thoroughly described in the WeCA Chapter 3. Post-crawl, the web pages underwent the cleaning and storage process before being divided into data chunks, as depicted in Chapter 4.

In the face of incomplete data within the initial set, our methodology involved filtering out underrepresented categories. Consequently, the final dataset incorporates only those categories with at least three websites, amounting to a total of 60 individual categories. These categories were further classified into 9 overarching themes based on the client's specifications. The distribution and structure of these themes are restricted by the client's criteria, which imposed a specific framework on our dataset categorization. Thus, we introduced three types of datasets for our experiments. First, we use the whole dataset based on its categories to which we will refer to it as FDC, secondly we do the same as the latter but this time we consider the themes as the categories, this dataset is referred to as FDTC. Lastly, we divide the whole

[3] www.olfeo.fr

dataset to generate subset datasets referring to each theme in our whole dataset.

The dataset, as summarized in Table 5.1, encompasses 6592 websites across the defined categories. These websites collectively consist of 131,553 web pages, which are further segmented into 722,003 discrete chunks of data. This delineation is not merely quantitative; it represents a systematically categorized dataset that underpins the experimental inquiries set forth in this experiment. The big data size of this dataset thus presents both a challenge and an opportunity to test the methods developed in Chapters 3 and 4. For a more detailed view of the dataset see Table 5.2.

Table 5.1: Big Dataset Summary

Theme Label	Count Categories	Websites	Web pages	Chunks
Legal Risk	6	402 (6,10%)	10337 (7,86%)	32037 (4,44%)
Security Risk	3	24 (0,36%)	162 (0,12%)	2116 (0,29%)
Adult Content	4	33 (0,50%)	345 (0,26%)	4555 (0,63%)
Risk of Unauthorized Disclosure	5	28 (0,42%)	200 (0,15%)	2571 (0,36%)
Bandwidth	5	213 (3,23%)	2318 (1,76%)	30875 (4,28%)
Leisure and Social Activities	19	2294 (34,80%)	52447 (39,87%)	266672 (36,94%)
Education	3	566 (8,59%)	11939 (9,08%)	49380 (6,84%)
Consumer Services	9	2000 (30,34%)	32335 (24,58%)	226018 (31,30%)
Business Services	6	1032 (15,66%)	21470 (16,32%)	107779 (14,93%)
Sum	**60**	**6592**	**131553**	**722003**

Table 5.1 provides a comprehensive summary of FDC categorized into various themes. Each theme is represented by its label, indicating different content categories, along with corresponding counts across websites, web pages, and text chunks. In the other hand, Table 5.2 offers a detailed breakdown of FDC categorized into various themes and subcategories. Each theme is presented alongside its corresponding categories, showcasing the distribution of websites, web pages, and text chunks within each category.

5.4 Experiments

The technical foundation of our experiments relies on Python 3.10, Docker for containerization, Huggingface framework for natural language processing, MLflow for machine learning lifecycle management, and BentoML for model serving. Version control and automated deployment are facilitated by a GitLab CI/CD pipeline. The computational demands are met by a cloud service equipped with up to four A100 GPUs.

In our experiments, we utilize a variety of models to tackle the complexity of web content classification. Notably, the XLM-R (XLM-RoBERTa) model, in both its 'large' and 'base' configurations, stood out due to its extensive training on multilingual datasets, making it a cornerstone of our analysis. This model harnesses the power of cross-lingual representation, making it especially adept at processing and understanding a wide range of languages and dialects. Furthermore, the mBERT (Multilingual BERT) model, available in both uncased and cased versions, was evaluated to understand the influence of case sensitivity on classification performance. These models are among the most sophisticated in the field of NLP, providing nuanced insights into language and offering robust solutions for navigating the vast terrain of web data.

In the following, we present an analysis of the performance of advanced language models applied to the classification of a large-scale dataset. This analysis includes a comparison of accuracy and weighted F-scores for models across both categorical and thematic classifications. We evaluate the models effectiveness in handling diverse web content and highlight significant patterns and discrepancies in their performances. The discussion aims to interpret these outcomes, offering insights into the practical implications for the application of natural language processing techniques in web content analysis. This thorough evaluation contributes to the broader understanding of the strengths and limitations of current language models for our task of website classification applied in the big data context.

Concurrently, we continuously evaluated the performance of multilingual LLMs, focusing on their ability to adapt to and process content across various languages within the dataset, identifying operational strengths and weaknesses. All results are presented in Tables 5.3 and 5.4.

Table 5.3: Summary of all Experiments Accuracies

Dataset	Label	RoBERTa	mBERT$_{base_{uncased}}$	mBERT$_{base_{cased}}$	XLM-R$_{large}$	XLM-R$_{base}$
FDC	/	65,68 %	69,88 %	69,38 %	72,09 %	**75,58 %**
FDTC	/	73,43 %	79,12 %	77,00 %	81,33 %	**83,01 %**
Theme 1	**Legal Risk**	90,32 %	92,00 %	92,69 %	**96,35 %**	**96,85 %**
Theme 2	**Security Risk**	91,73 %	93,27 %	91,12 %	**96,34 %**	**96,67 %**
Theme 3	**Adult Content**	92,22 %	95,76 %	90,98 %	**90,09 %**	**97,60 %**
Theme 4	**Risk of Unauthorized Disclosure**	64,50 %	66,34 %	66,90 %	67,24 %	**71,03 %**
Theme 5	**Bandwidth**	83,74 %	86,09 %	86,06 %	**85,54 %**	**90,58 %**
Theme 6	**Leisure and Social Activities**	71,60 %	74,40 %	73,62 %	**77,15 %**	76,89 %
Theme 7	**Education**	87,10 %	87,14 %	87,39 %	**88,32 %**	**91,55 %**
Theme 8	**Consumer Services**	80,92 %	81,01 %	80,37 %	**85,34 %**	**85,67 %**
Theme 9	**Business Services**	74,00 %	78,07 %	79,04 %	80,87 %	**82,18 %**

Table 5.4: Summary of all Experiments Weighted F-Scores

Dataset	Label	RoBERTa	mBERT$_{base_{uncased}}$	mBERT$_{base_{cased}}$	XLM-R$_{large}$	XLM-R$_{base}$
FDC	/	64,54 %	68,98 %	67,17 %	69,63 %	**74,45 %**
FDTC	/	71,35 %	77,49 %	75,11 %	80,37 %	**81,27 %**
Theme 1	**Legal Risk**	88,37 %	89,96 %	91,98 %	**93,91 %**	**95,78 %**
Theme 2	**Security Risk**	90,49 %	90,78 %	89,58 %	**95,01 %**	**95,46 %**
Theme 3	**Adult Content**	91,15 %	93,88 %	88,76 %	**87,87 %**	**95,86 %**
Theme 4	**Risk of Unauthorized Disclosure**	62,16 %	65,57 %	65,24 %	66,49 %	**70,04 %**
Theme 5	**Bandwidth**	82,92 %	84,86 %	85,12 %	**84,30 %**	**88,44 %**
Theme 6	**Leisure and Social Activities**	69,88 %	73,23 %	71,71 %	**75,20 %**	74,53 %
Theme 7	**Education**	85,43 %	86,40 %	85,99 %	**86,57 %**	**90,01 %**
Theme 8	**Consumer Services**	78,43 %	80,03 %	78,02 %	**84,54 %**	**83,92 %**
Theme 9	**Business Services**	72,38 %	75,78 %	78,28 %	78,82 %	**80,38 %**

5.4.1 Comparative Analysis of Classifiers through FDC and FDTC

The first phase of our experimentation involve a category-based classification of the entire dataset to determine the framework's effectiveness in categorizing web content into predefined categories, evaluating precision and recall rates across diverse themes. Additionally, we expanded our scope by treating themes as categories themselves, applying our models to the entire dataset

under this broader schema. This aimed to observe how model performance shifted when transitioning from specific categories to more general themes.

In **FDC**, classifiers are scrutinized for their ability to discriminate between numerous finely divided categories:

- **RoBERTa** lags with an accuracy of 65.68% and a weighted F-score of 64.54%. These figures indicate that while RoBERTa can handle individual categories, its performance is not optimal in a multifaceted setting.
- **mBERT Variants** show an improvement in both metrics over RoBERTa, with the uncased variant displaying an accuracy of 69.88% and a weighted F-score of 68.98%. The cased variant has similar accuracy at 69.38% but a slightly lower F-score at 67.17%, which could imply that the uncased variant has a better balance of precision and recall in this context.
- **XLM-R Models** outperform the field, particularly the base model with the highest accuracy of 75.58% and a weighted F-score of 74.45%. These scores suggest that the XLM-R base is most attuned to the task's demands, successfully identifying category-specific linguistic patterns.

FDTC provides a view of how models manage broader content groupings, where themes may present more generalized linguistic features:

- **RoBERTa** shows a significant performance boost in FDTC, with an increase in accuracy to 73.43% and a weighted F-score of 71.35%. This suggests that RoBERTa is more adept at theme-level generalization than fine-grained category differentiation.
- **mBERT Variants** see similar improvements, with the uncased model reaching an accuracy of 79.12% and a weighted F-score of 77.49%. The cased variant also shows gains but remains slightly behind its uncased counterpart, underscoring the potential impact of case sensitivity on thematic classification.
- **XLM-R Models** continue to lead, with the base variant achieving an

accuracy of 83.01% and a weighted F-score of 81.27% in FDTC. The large variant also performs strongly, with an accuracy of 81.33% and a weighted F-score of 80.37%. The superior performance of XLM-R models in the thematic dataset underscores their robust capacity for capturing and generalizing across diverse linguistic landscapes.

When comparing FDC and FDTC, it's evident that all classifiers demonstrate enhanced performance metrics in the FDTC, suggesting that thematic classification poses less complexity and challenges for these models compared to detailed categorical classification. This improvement is reflected in both accuracy and weighted F-scores. Particularly noteworthy is the increase in weighted F-scores from FDC to FDTC, especially for RoBERTa and mBERT models. This enhancement might be attributed to a clearer thematic distinction aligning well with the pre-training corpora of these models. Moreover, the consistent robustness of XLM-R models across both datasets underscores their versatility in seamlessly transitioning between detailed and generalized classification tasks without any significant loss in performance.

5.4.2 Comparative Analysis of Classifiers on Theme-Based Datasets

The second experimental phase split the dataset into sub-datasets corresponding to each theme, allowing for an in-depth analysis of the model's capabilities with theme-segregated content. This provided insights into the specificity and adaptability of the models to theme-focused classification challenges.

The results from both accuracy and weighted F-scores provide a nuanced view of how each classifier deals with the intricacies of different thematic contents.

- **Legal Risk (Theme 1)**: A domain requiring precision, where the XLM-R models, particularly the base variant with an accuracy of 96.85% and a weighted F-score of 95.78%, show outstanding ability to capture the formal register and specialized lexicon.

- **Security Risk (Theme 2)**: Similar to Theme 1, this theme demands a grasp of specialized terminology, which is well-met by XLM-R models, with the base variant reaching an accuracy of 96.67% and a weighted F-score of 95.46%, suggesting adeptness in handling security-related content.
- **Adult Content (Theme 3)**: Here, the nuanced understanding of social and cultural contexts is critical. The XLM-R base model's high accuracy (97.60%) and weighted F-score (95.86%) suggest superior contextual sensitivity and content discernment.
- **Risk of Unauthorized Disclosure (Theme 4)**: The lowest scores across all themes for every model, with XLM-R base having the highest weighted F-score of 70.04%, indicate a universal challenge posed by this theme, which may require models to have more nuanced training on privacy-related content.
- **Bandwidth (Theme 5)**: A technical theme where all models perform relatively well, but the XLM-R base still leads with an accuracy of 90.58% and a weighted F-score of 88.44%, indicating its effectiveness in technical domains.
- **Leisure and Social Activities (Theme 6)**: Showcases the challenge of informal, varied content. The XLM-R large performs the best among models, with a balanced approach to a diversity of linguistic expressions.
- **Education (Theme 7)**: Reflects content with a range of complexity, from academic to general knowledge. Here, XLM-R base scores an accuracy of 91.55% and a weighted F-score of 90.01%, showcasing its capability in processing educational material.
- **Consumer Services (Theme 8)**: Encompasses a variety of language uses, from marketing to service feedback, where XLM-R models show higher scores, indicating their flexibility and broader understanding of consumer discourse.
- **Business Services (Theme 9)**: The professional and technical lan-

guage of this theme is best captured by the XLM-R base, with an accuracy of 82.18% and a weighted F-score of 80.38%, suggesting that XLM-R models are well-suited for business-related content.

The performance of models across thematic datasets reveals a distinct pattern in the ability of various classifiers to handle thematic content. XLM-R models consistently demonstrate high performance, which emphasizes their strong capability to generalize and adapt across a range of thematic contents. This adaptability is a significant asset, as it suggests that XLM-R models are adept at managing the diversity inherent in thematic datasets.

mBERT models, including both the uncased and cased variants, also show strong performance. However, there is noticeable variability in their effectiveness across different themes. This variation could be indicative of the models' sensitivity to the linguistic nuances and the specific demands of each domain. Such sensitivity implies that while mBERT models are generally capable, their performance might be optimized by tuning them to the specific linguistic and conceptual characteristics of each theme.

RoBERTa, on the other hand, exhibits a lower performance in category-based classification but shows improvement when applied to thematic contexts. Nonetheless, it does not reach the benchmark set by the XLM-R models. The variation in RoBERTa's performance across different themes could be attributed to its pre-training corpus and model architecture. These aspects of RoBERTa might be less suited for the task of thematic generalization when compared to the XLM-R models, suggesting a potential area for further model refinement to improve its adaptability to thematic content.

5.4.3 Performance Analysis

The analysis of performance across the classifiers provides insights into several key aspects that determine the success of language models in web content classification tasks. A critical factor is the scope and diversity of the training data, particularly for multilingual models. The XLM-R's exceptional train-

ing on a multilingual dataset is a contributing factor to its distinguished performance. Its capability to process and understand various contexts in multiple languages grants it a notable advantage, especially in Whole Dataset Themes as Categories (FDTC), where themes can manifest a broader spectrum of linguistic diversity.

Additionally, the investigation into the effects of case sensitivity on model performance has yielded some enlightening observations. The consistent outperformance of the mBERT (base uncased) model compared to its cased counterpart has implications for the development of NLP applications. This uncased model's ability to lead suggests that the presence of case information in text data may not significantly enhance the model's ability to comprehend the context within web content classification, indicating potential benefits in ignoring case.

Lastly, when examining the granularity of content categorization against the ability to generalize, the XLM-R base model stands out. It not only navigates the fine-grained distinctions within the Whole Dataset Category-based (FDC) effectively but also demonstrates an exceptional capacity for generalizing in FDTC. This versatility is paramount for classifiers that are expected to contend with the challenging and varied landscape of web content, adapting their approach to match the level of specificity required by the task at hand. The leading performance of the XLM-R base underscores its readiness for a wide array of classification challenges, from the detailed to the thematic.

5.4.4 XLM-R and Fine-tuning Benchmark

Evaluating fine-tuning techniques for pre-trained language models is crucial for optimizing model performance, managing computational resources, and enhancing the overall understanding of model adaptation strategies. Standard fine-tuning, while effective, demands significant computational power, making techniques like LoRA, QLoRA, and adapters valuable alternatives.

These methods offer efficient parameter updates, reducing training costs while achieving competitive performance. Evaluating these techniques helps identify the most suitable approach for specific tasks and datasets, ensuring models are both effective and resource-efficient.

In this section, we detail the fine-tuning techniques employed in our experiments, focusing on Standard Fine-Tuning, Low-Rank Adaptation (LoRA), Quantized LoRA (QLoRA), and Adapter-based methods. Each technique is applied to the XLM-RoBERTa base model to evaluate their performance on the website classification tasks. Table 5.5 summarize the pros and cons of these three techniques. Table 5.6 summarize the obtained weighted F-score results of this benchmark.

Standard fine-tuning involves updating all the parameters of the pre-trained model during the training process. This method leverages the entire capacity of the model to adapt to the specific downstream task. By fine-tuning all layers, the model learns task-specific features that enhance its performance on the given dataset.

Low-Rank Adaptation (LoRA) is a parameter-efficient adaptation method that introduces low-rank matrices into the existing model layers. Instead of fine-tuning all the model parameters, LoRA updates only the introduced low-rank matrices, significantly reducing the number of trainable parameters and computational overhead. This method is particularly useful for scenarios with limited computational resources while still achieving competitive performance [329].

Quantized LoRA (QLoRA) builds on the LoRA approach by further quantizing the low-rank matrices. This quantization reduces the memory footprint and computational requirements even more than LoRA, making it suitable for deployment in resource-constrained environments. Despite the reduced precision, QLoRA maintains a performance close to full-precision models, making it an attractive option for practical applications [330].

Adapters methods involve adding small neural network modules (adapters)

between the layers of the pre-trained model. During fine-tuning, only the parameters of these adapters are updated while keeping the original model parameters fixed. This approach drastically reduces the number of trainable parameters and allows for efficient task adaptation. Adapters have been shown to be effective in multi-task learning and transfer learning scenarios [331].

Table 5.5: Pros and Cons of Fine-Tuning Techniques

Technique	Pros	Cons
Standard Fine-Tuning	• Full model adaptation • Best performance	• High computational cost • Requires significant resources
LoRA	• Reduced number of trainable parameters • Computationally efficient	• Slightly lower performance • Additional complexity in implementation
QLoRA	• Further reduces memory footprint • Suitable for resource-constrained environments	• Potential performance drop due to quantization • Requires careful tuning
Adapters	• Efficient task adaptation • Drastically reduces trainable parameters	• Typically lower performance than full fine-tuning • May not generalize as well to all tasks

Table 5.2: Detailed Whole Dataset Themes and Categories

Theme	Category	Websites	Web pages	Chunks
1	**Counterfeiting and Copyright Infringement**	150 (37,31%)	3748 (36,26%)	8865 (27,67%)
	Illicit Drugs	14 (3,48%)	100 (0,97%)	2137 (6,67%)
	Gambling	17 (4,23%)	67 (0,65%)	1530 (4,78%)
	Pirated Music, Films, Software	17 (4,23%)	53 (0,51%)	2297 (7,17%)
	Terrorism, Violence, Explosives and Poisons	19 (4,73%)	165 (1,60%)	1073 (3,35%)
	Escort Services	185 (46,02%)	6204 (60,02%)	16135 (50,36%)
2	**Proxies, Redirectors**	7 (29,17%)	38 (23,46%)	502 (23,72%)
	Parked Domains	8 (33,33%)	103 (63,58%)	1364 (64,46%)
	URL Reducers	9 (37,50%)	21 (12,96%)	250 (11,81%)
3	**Sex, Pornography**	8 (24,24%)	51 (14,78%)	1810 (39,74%)
	Weapons, Hunting, Safety Equipment	8 (24,24%)	135 (39,13%)	1037 (22,77%)
	Dating Services	8 (24,24%)	87 (25,22%)	930 (20,42%)
	Alcohol and Tobacco	9 (27,27%)	72 (20,87%)	778 (17,08%)
4	**Web Hosting, ISP**	3 (10,71%)	35 (17,50%)	109 (4,24%)
	Personal Pages	9 (32,14%)	9 (4,50%)	270 (10,50%)
	Blogs	9 (32,14%)	9 (4,50%)	1719 (66,86%)
	Forums, Wikis	3 (10,71%)	22 (11,00%)	400 (15,56%)
	Social Networks	4 (14,29%)	125 (62,50%)	73 (2,84%)
5	**"Get paid to surf" Programs**	17 (7,98%)	352 (15,19%)	532 (1,72%)
	Television	50 (23,47%)	770 (33,22%)	10432 (33,79%)
	Video Sharing	46 (21,60%)	44 (1,90%)	4916 (15,92%)
	Website Analytics	50 (23,47%)	769 (33,18%)	4753 (15,39%)
	Radio	50 (23,47%)	383 (16,52%)	10242 (33,17%)
6	**Traditional Religions**	250 (10,90%)	787 (1,50%)	10626 (3,98%)
	Photography, Image Databases	50 (2,18%)	795 (1,52%)	10361 (3,89%)
	Cars, Motor Sports	194 (8,46%)	6481 (12,36%)	18572 (6,96%)
	Sports	60 (2,62%)	879 (1,68%)	12030 (4,51%)
	Cinema	50 (2,18%)	730 (1,39%)	9705 (3,64%)
	Weather	50 (2,18%)	444 (0,85%)	5630 (2,11%)
	Mobile Phones, Logos, Ringtones	50 (2,18%)	815 (1,55%)	10427 (3,91%)
	Fashion, Beauty, Wellness	196 (8,54%)	7261 (13,84%)	17917 (6,72%)
	Tourism, Hotels	756 (32,96%)	20051 (38,23%)	51300 (19,24%)
	Art and Culture	50 (2,18%)	823 (1,57%)	10855 (4,07%)
	Social Issues	50 (2,18%)	743 (1,42%)	10207 (3,83%)
	Media, News	50 (2,18%)	1210 (2,31%)	16472 (6,18%)
	Online Shopping	50 (2,18%)	896 (1,71%)	11701 (4,39%)
	IT and Computing	50 (2,18%)	918 (1,75%)	11739 (4,40%)
	Video Games, Online Games	50 (2,18%)	884 (1,69%)	11792 (4,42%)
	Celebrities	50 (2,18%)	745 (1,42%)	10285 (3,86%)
	Restoration	50 (2,18%)	815 (1,55%)	11140 (4,18%)
	Pets and Animals	50 (2,18%)	738 (1,41%)	10358 (3,88%)
	Home, Garden, Interior Decorating	188 (8,20%)	6432 (12,26%)	15555 (5,83%)
7	**Science, Research**	134 (23,67%)	944 (7,91%)	12905 (26,13%)
	Education	367 (64,84%)	10327 (86,50%)	28010 (56,72%)
	Childhood	65 (11,48%)	668 (5,60%)	8465 (17,14%)
8	**Real Estate**	155 (7,75%)	2299 (7,11%)	30704 (13,58%)
	Jobs	155 (7,75%)	2276 (7,04%)	30358 (13,43%)
	Investment, Personal Finance	155 (7,75%)	2302 (7,12%)	31103 (13,76%)
	Classified Ads	147 (7,35%)	1221 (3,78%)	16813 (7,44%)
	Banks, Insurance, Social Benefits	543 (27,15%)	13623 (42,13%)	37106 (16,42%)
	Health	154 (7,70%)	2387 (7,38%)	32260 (14,27%)
	Consumer Services	202 (10,10%)	5696 (17,62%)	14624 (6,47%)
	Search Engines, Portals	464 (23,20%)	2315 (7,16%)	30144 (13,34%)
	Online Auctions	25 (1,25%)	216 (0,67%)	2906 (1,29%)
9	**Law, Taxes**	77 (7,46%)	1299 (6,05%)	17089 (15,86%)
	Government	172 (16,67%)	1345 (6,26%)	17566 (16,30%)
	Business Services	587 (56,88%)	16918 (78,80%)	47658 (44,22%)
	Business Forums, Wikis	86 (8,33%)	1180 (5,50%)	15940 (14,79%)
	Translation	86 (8,33%)	146 (0,68%)	1937 (1,80%)
	Guides, Maps, Traffic Conditions	24 (2,33%)	582 (2,71%)	7589 (7,04%)

Table 5.6: Weighted F-Scores for the Fine-Tuning Techniques

Dataset	Label	Standard Fine-Tuning	LoRA	QLoRA	Adapters
FDC	/	**75,58 %**	73,21 %	73,63 %	69,11 %
FDTC	/	**83,01 %**	80,52 %	81,23 %	78,74 %
Theme 1	**Legal Risk**	**96,85 %**	93,45 %	93,16 %	89,54 %
Theme 2	**Security Risk**	**96,67 %**	94,64 %	93,27 %	90,93 %
Theme 3	**Adult Content**	**97,60 %**	93,32 %	91,78 %	91,14 %
Theme 4	**Risk of Unauthorized Disclosure**	**71,03 %**	68,01 %	68,72 %	65,21 %
Theme 5	**Bandwidth**	**90,58 %**	88,98 %	88,52 %	87,96 %
Theme 6	**Leisure and Social Activities**	**76,89 %**	73,34 %	74,09 %	72,32 %
Theme 7	**Education**	**91,55 %**	88,27 %	85,83 %	86,02 %
Theme 8	**Consumer Services**	**85,67 %**	84,39 %	83,76 %	80,12 %
Theme 9	**Business Services**	**82,18 %**	81,01 %	80,48 %	79,77 %

Discussion

Standard fine-tuning consistently achieves the highest weighted F-scores across all datasets. This method involves updating all model parameters, allowing the model to fully adapt to the specific task. The scores range from 71.03% for "Risk of Unauthorized Disclosure" to 97.60% for "Adult Content," showcasing the effectiveness of this technique in utilizing the model's full capacity for different types of classification tasks. This indicates that standard fine-tuning, while resource-intensive, maximizes the model's performance by making full use of its parameters.

LoRA generally performs well but shows a slight decrease in performance compared to standard fine-tuning. The weighted F-scores range from 68.01% to 94.64%, indicating that while LoRA is efficient in reducing the number of trainable parameters, it does come at a cost of reduced performance. For instance, in the "Adult Content" category, LoRA achieves 93.32%, which is noticeably lower than the 97.60% achieved by standard fine-tuning. This suggests that while LoRA is useful for scenarios with limited computational resources, it may not always match the highest performance levels of full fine-tuning.

QLoRA shows competitive performance with LoRA, with weighted F-scores ranging from 68.72% to 93.27%. This method combines the benefits of LoRA with quantization to further reduce memory and computational requirements. Despite these efficiencies, the slight performance drop compared to standard fine-tuning and LoRA suggests that the quantization process may introduce some loss in precision, affecting the overall model performance. Nevertheless, QLoRA remains a strong candidate for resource-constrained environments, offering a good balance between efficiency and performance.

Adapter-based fine-tuning demonstrates the lowest performance among the evaluated techniques, with weighted F-scores ranging from 65.21% to 91.14%. This method involves updating only small neural network modules added to the pre-trained model layers, significantly reducing the number of trainable parameters. While adapters offer efficient task adaptation, the reduced num-

ber of trainable parameters appears to limit their performance compared to the other techniques. For example, the score for "Legal Risk" is 89.54%, which is lower than the scores achieved by both LoRA and QLoRA. This indicates that adapters, while efficient, may not be as effective for tasks requiring comprehensive model adaptation.

In summary, the results indicate that while standard fine-tuning yields the highest performance, it requires significant computational resources. LoRA and QLoRA provide efficient alternatives with competitive performance, suitable for scenarios where computational resources are limited. Adapter-based fine-tuning, although less effective in this comparison, still offers a viable solution for multi-task learning and transfer learning scenarios, particularly when resource efficiency is paramount. Overall, the choice of fine-tuning technique should consider the trade-off between performance and computational efficiency, tailored to the specific requirements and constraints of the application at hand.

5.5 Conclusion

In this chapter, we have systematically approached the classification of a large-scale, multilingual dataset, deploying a suite of sophisticated language models to parse and categorize web content. Our experiments and findings have significantly contributed to the understanding of how different models handle the vast and varied landscape of web data, offering insights into the biases inherent in web dynamics.

Through rigorous experimentation, we have demonstrated that the dichotomy of data — whether treated categorically or thematically — can be effectively navigated using current language models, specifically the XLM-R and mBERT variants. The strategic division of the dataset enabled us to mitigate the challenges posed by the big data characteristics of volume, variety, and velocity, thus addressing concerns of web dynamics bias. The thematic approach, in particular, proved to be a robust method in conquering these

challenges, providing a more coherent and generalizable basis for classification than granular categories.

Our research has laid a foundation for the application of dynamic and distributed classifier combination techniques, such as those found in Multi-agent systems, which could further enhance the efficiency and accuracy of web content classification. This approach could leverage the strengths of various models, distributing tasks to optimize performance and adaptability.

Looking forward, future research should explore the benchmarking of newly open-sourced Large Language Models (LLMs) and their direct application to our use case. There is significant potential for these models to offer improved understanding and interpretation of web content, especially when applied in a distributed manner that captures the complex, dynamic nature of web data.

In conclusion, the methodologies and findings detailed in this chapter provide a valuable contribution to the field, expanding the toolkit for tackling web dynamics bias and setting a course for future research in big data classification. The continuous evolution of language models and classification techniques promises further advancements, guiding us toward more nuanced and sophisticated approaches to understanding the web's intricate fabric.

Chapter 6

Conclusion

6.1 Introduction

This thesis has systematically tackled the substantial challenges associated with website and web page classification, addressing the evolving needs of an era dominated by digital data proliferation. As the internet continues to expand, the volume and variety of content demand advanced and efficient classification systems to enhance accessibility, relevance, and security of information. This research journey has encompassed a comprehensive approach that includes in-depth reviews of existing methods, development of new methodologies, and their application in practical settings to improve text classification and optimize web content categorization.

The importance of this work lies not only in its contribution to academic knowledge but also in its practical implications for industries ranging from digital marketing to cybersecurity. By enhancing the precision and efficiency of classification systems, this thesis supports the broader goal of making web content more navigable and useful for various stakeholders, including content creators, marketers, and end-users.

Through an examination of traditional, neural network-based, and hybrid methodologies, this thesis has provided clarity on the evolution of text classification techniques. It has highlighted the transition from foundational statistical methods, which laid the groundwork for initial classification ef-

forts, to more sophisticated neural network models that offer deeper insights into content semantics and user context. This progression has been instrumental in developing the Website Classification Approach (WeCA) and the Weighted Stratified Split Approach (WSSA), both of which signify major advancements in the field.

6.2 Contributions

1. **Survey on Text Representation Techniques and Web Classification Approaches:** The thesis provides a detailed survey of text representation techniques and web page and website classification approaches in Chapter 2. This study offers a foundational understanding of the methods evolution and current state, serving as a resource for researchers and practitioners.

2. **Development and Validation of the Website Classification Approach (WeCA):** The WeCA methodology, documented in the Chapter 3 and also published in [332]: *M. Zohir Koufi, Z. Guessoum, A. Keziou, I. Yahiaoui, "Toward Website Classification," 2023 IEEE/WIC International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), Venice, Italy, 2023, pp. 306-310, doi: 10.1109/WI-IAT59888.2023.00049, advances website classification.* This approach combines techniques into a hybrid model that shows improved accuracy and adaptability for web content classification.

3. **Text Chunking with the Weighted Stratified Split Approach (WSSA):** The thesis introduced WSSA to improve website classification, particularly for long texts. This method is detailed in Chapter 4 and also published in [333] : *Koufi, M.Z., Guessoum, Z., Keziou, A., Yahiaoui, I. (2024). Text Chunking to Improve Website Classification. In: Pereira, A.I., Mendes, A., Fernandes, F.P., Pacheco, M.F., Coelho, J.P., Lima, J. (eds) Optimization, Learning Algorithms and Applications. OL2A 2023. Communications in Computer and Infor-*

mation Science, vol 1981. Springer, Cham. https://doi.org/10.1007/978-3-031-53025-8_15.

4. **Scalability and System Implementation for a Larger Dataset:** The scalability of the WeCA and WSSA methodologies and their application to a larger dataset led to the implementation of a system for Olfeo (Chapter 5). This application showed the methods effectiveness in real-world settings. The use of multilingual LLMs and dichotomy strategies addressed scalability challenges in web content classification, enabling the handling of diverse and large web content.

6.3 Perspectives

Future research in website and web page classification can expand in several directions:

1. **Exploration of Deep Learning Architectures:** Investigation into deep learning architectures can improve semantic understanding and contextual analysis of web content.
2. **Integration with Emerging Technologies:** Converging NLP and ML with technologies like blockchain and quantum computing could lead to more secure and efficient web content classification systems.
3. **Expansion to Multimodal Contexts:** Extending methodologies to handle multimodal data (text, images, video) will address comprehensive classifications across media types, building on the success of multilingual LLMs.
4. **Dynamic Multi-Agent Based Solutions in Collaborative Approaches:** Adopting dynamic multi-agent-based solutions will facilitate responsive and adaptive collaborative approaches, enhancing the system's ability to evolve in complex environments.
5. **Implementation of a RAG System for Better Evolution:** In-

troducing a RAG system will provide a structured way to evaluate and refine the classification solution, ensuring adaptability.

6. **Use of Vector Databases:** Implementing vector databases will support efficient storage and retrieval of complex data structures, enhancing the performance of the classification system.

7. **Focus on Personalization and User-Centric Models:** Developing personalized and user-centric classification models using user feedback and adaptive learning mechanisms can improve user engagement and satisfaction.

8. **Ethical Considerations and Bias Mitigation:** Addressing ethical considerations and mitigating biases in algorithms is essential to ensure transparent and fair systems.

In conclusion, this thesis has established a path for exploration and development in the field of website and web page classification. The insights gained, methodologies developed, and systems implemented pave the way for future innovation and research to effectively meet the digital age challenges.

Bibliography

[1] Bing Liu. *Sentiment analysis and opinion mining.* Springer Nature, 2022.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[3] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11): 613–620, 1975.

[4] Ian T Jolliffe. *Principal component analysis for special types of data.* Springer, 2002.

[5] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[7] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL `https://aclanthology.org/D14-1162`.

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

[9] Yoon Kim. Convolutional neural networks for sentence classification. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, oct 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1181. URL `https://aclanthology.org/D14-1181`.

[10] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[12] Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, K. Meimandi, Matthew S. Gerber, and Laura E. Barnes. Hdltex: Hierarchical deep learning for text classification. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 364–371, 2017. URL `https://api.semanticscholar.org/CorpusID:393535`.

[13] Kamran Kowsari, Mojtaba Heidarysafa, Donald E. Brown, Kiana Jafari Meimandi, and Laura E. Barnes. Rmdl: Random multimodel deep learning for classification. In *Proceedings of the 2nd International Conference on Information System and Data Mining*, ICISDM '18, page 19–28, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450363549. doi: 10.1145/3206098.3206111. URL `https://doi.org/10.1145/3206098.3206111`.

[14] Hao Peng, Jianxin Li, Senzhang Wang, Lihong Wang, Qiran Gong,

Renyu Yang, Bo Li, S Yu Philip, and Lifang He. Hierarchical taxonomy-aware and attentional graph capsule rcnns for large-scale multi-label text classification. *IEEE Transactions on Knowledge and Data Engineering*, 33(6):2505–2519, 2019.

[15] Aurangzeb Khan, Baharum Baharudin, Lam Hong Lee, and Khairullah Khan. A review of machine learning algorithms for text-documents classification. *Journal of advances in information technology*, 1(1): 4–20, 2010.

[16] Bhat S Harish, Devanur S Guru, and Shantharamu Manjunath. Representation and classification of text documents: A brief review. *IJCA, Special Issue on RTIPPR (2)*, 110:119, 2010.

[17] Ani Nenkova and Kathleen McKeown. A survey of text summarization techniques. *Mining text data*, pages 43–76, 2012.

[18] Sheetal S Sonawane and Parag A Kulkarni. Graph based representation and analysis of text document: A survey of techniques. *International Journal of Computer Applications*, 96(19), 2014.

[19] Lili Wang, Chongyang Gao, Jason Wei, Weicheng Ma, Ruibo Liu, and Soroush Vosoughi. An empirical survey of unsupervised text representation methods on twitter data. *arXiv preprint arXiv:2012.03468*, 2020.

[20] Riad Sonbol, Ghaida Rebdawi, and Nada Ghneim. The use of nlp-based text representation techniques to support requirement engineering tasks: A systematic mapping review. *IEEE Access*, 2022.

[21] Phu Pham, Loan TT Nguyen, Witold Pedrycz, and Bay Vo. Deep learning, graph-based text representation and classification: a survey, perspectives and challenges. *Artificial Intelligence Review*, 56(6):4893–4927, 2023.

[22] Rajvardhan Patil, Sorio Boit, Venkat Gudivada, and Jagadeesh Nandigam. A survey of text representation and embedding techniques

in nlp. *IEEE Access*, 11:36120–36146, 2023. doi: 10.1109/ACCESS.2023.3266377.

[23] Gerald Gazdar. *Generalized phrase structure grammar*. Harvard University Press, 1985.

[24] Richard A Sharman, Frederick Jelinek, and Robert L Mercer. Generating a grammar for statistical training. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990.

[25] Peter RJ Asveld. Fuzzy context-free languages—part 2: Recognition and parsing algorithms. *Theoretical computer science*, 347(1-2):191–213, 2005.

[26] Fangju Wang. A fuzzy grammar and possibility theory-based natural language user interface for spatial queries. *Fuzzy sets and systems*, 113 (1):147–159, 2000.

[27] Rajvardhan Patil and Zhengxin Chen. Struct: Incorporating contextual information for english query search on relational databases. In *Proceedings of the Third International Workshop on Keyword Search on Structured Data*, pages 11–22, 2012.

[28] Rajvardhan Patil, Zhengxin Chen, and Yong Shi. Database keyword search: a perspective from optimization. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 30–33. IEEE, 2012.

[29] Rajvardhan Patil, Sorio Boit, and Nathaniel Bowman. Sql chatbot–using context free grammar. In *2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, pages 1–7. IEEE, 2022.

[30] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[31] Slava Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401, 1987.

[32] Si Lhoussain Aouragh, Abdellah Yousfi, Saida Laaroussi, Hicham Gueddah, and Mohammed Nejja. A new estimate of the n-gram language model. *Procedia Computer Science*, 189:211–215, 2021. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2021.05.111. URL `https://www.sciencedirect.com/science/article/pii/S1877050921012382`. AI in Computational Linguistics.

[33] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999.

[34] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Frederick Jelinek, John Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85, 1990.

[35] Rajvardhan Patil and Asim Shrestha. Feature-set for sentiment analysis. In *2019 SoutheastCon*, pages 1–5. IEEE, 2019.

[36] Clayton Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 216–225, 2014.

[37] Finn Årup Nielsen. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*, 2011.

[38] Saif Mohammad and Peter Turney. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 26–34, 2010.

[39] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, 2004.

[40] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pages 347–354, 2005.

[41] Gerard Salton and Michael E Lesk. Computer evaluation of indexing and text processing. *Journal of the ACM (JACM)*, 15(1):8–36, 1968.

[42] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 60(5):493–502, 2004.

[43] G Wael. a.(2013). a survey of text similarity approaches. *International Jisuanji Yingyong*, pages 13–18.

[44] Giambattista Amati. *BM25*, pages 257–260. Springer US, Boston, MA, 2009. ISBN 978-0-387-39940-9. doi: 10.1007/978-0-387-39940-9_921. URL `https://doi.org/10.1007/978-0-387-39940-9_921`.

[45] Kevin Lund and Curt Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior research methods, instruments, & computers*, 28(2):203–208, 1996.

[46] Curt Burgess, Kay Livesay, and Kevin Lund. Explorations in context space: Words, sentences, discourse. *Discourse Processes*, 25(2-3):211–257, 1998.

[47] Jacob Kogan, Charles Nicholas, and Vladimir Volkovich. Text mining with information-theoretic clustering. *Computing in Science & Engineering*, 5(6):52–59, 2003.

[48] Karl Pearson. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from

random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, July 1900. doi: 10.1080/14786440009463897. URL https://doi.org/10.1080/14786440009463897.

[49] Bin Tang, Michael Shepherd, Evangelos Milios, and Malcolm I Heywood. Comparing and combining dimension reduction techniques for efficient text clustering. In *Proceeding of SIAM international workshop on feature selection for data mining*, pages 17–26, 2005.

[50] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *Icml*, volume 97, page 35. Nashville, TN, USA, 1997.

[51] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[52] Susan T Dumais, George W Furnas, Thomas K Landauer, Scott Deerwester, and Richard Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–285, 1988.

[53] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine learning*, 42:177–196, 2001.

[54] Christian Jutten and Jeanny Herault. Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture. *Signal processing*, 24(1):1–10, 1991.

[55] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[56] Evgeniy Gabrilovich, Shaul Markovitch, et al. Computing semantic

relatedness using wikipedia-based explicit semantic analysis. In *IJcAI*, volume 7, pages 1606–1611, 2007.

[57] Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623*, 2014.

[58] Ben Athiwaratkun and Andrew Gordon Wilson. Multimodal word distributions. *arXiv preprint arXiv:1704.08424*, 2017.

[59] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[60] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[61] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. *Advances in neural information processing systems*, 30, 2017.

[62] Matthew E Peters, Mark Neumann, Robert L Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. Knowledge enhanced contextual word representations. *arXiv preprint arXiv:1909.04164*, 2019.

[63] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[64] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[65] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 51–61, 2016.

[66] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[67] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. *Advances in neural information processing systems*, 32, 2019.

[68] Billy Chiu, Gamal Crichton, Anna Korhonen, and Sampo Pyysalo. How to train good word embeddings for biomedical nlp. In *Proceedings of the 15th workshop on biomedical natural language processing*, pages 166–174, 2016.

[69] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017.

[70] Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual bert? *arXiv preprint arXiv:1906.01502*, 2019.

[71] Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. *Advances in neural information processing systems*, 32, 2019.

[72] Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, and Ming Zhou. Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks. *arXiv preprint arXiv:1909.00964*, 2019.

[73] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

[74] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*, 2019.

[75] Zewen Chi, Li Dong, Furu Wei, Wenhui Wang, Xian-Ling Mao, and Heyan Huang. Cross-lingual natural language generation via pre-training. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7570–7577, 2020.

[76] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*, 2019.

[77] Pei Ke, Haozhe Ji, Siyang Liu, Xiaoyan Zhu, and Minlie Huang. Sentilare: Sentiment-aware language representation learning with linguistic knowledge. *arXiv preprint arXiv:1911.02493*, 2019.

[78] Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2901–2908, 2020.

[79] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194, 2021.

[80] Zhiwei Jiang, Qing Gu, Yafeng Yin, and Daoxu Chen. Enriching word embeddings with domain knowledge for readability assessment. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 366–378, 2018.

[81] Kishlay Jha. Knowledge-base enriched word embeddings for biomedical domain. *arXiv preprint arXiv:2103.00479*, 2021.

[82] Bin Bi, Chen Wu, Ming Yan, Wei Wang, Jiangnan Xia, and Chenliang Li. Incorporating external knowledge into machine reading for generative question answering. *arXiv preprint arXiv:1909.02745*, 2019.

[83] Mehmet Umut Salur and Ilhan Aydin. A novel hybrid deep learning model for sentiment classification. *IEEE Access*, 8:58080–58093, 2020.

[84] Anwar Ur Rehman, Ahmad Kamran Malik, Basit Raza, and Waqar Ali. A hybrid cnn-lstm model for improving accuracy of movie reviews sentiment analysis. *Multimedia Tools and Applications*, 78:26597–26613, 2019.

[85] Aparna Khare, Srinivas Parthasarathy, and Shiva Sundaram. Multimodal embeddings using multi-task learning for emotion recognition. *arXiv preprint arXiv:2009.05019*, 2020.

[86] Farhad Nooralahzadeh, Lilja Øvrelid, and Jan Tore Lønning. Evaluation of domain-specific word embeddings using knowledge resources. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*, 2018.

[87] Steve Aibuedefe Aigbe and Christoph Eick. Learning domain-specific word embeddings from covid-19 tweets. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 4307–4312. IEEE, 2021.

[88] Timothy L Chen, Max Emerling, Gunvant R Chaudhari, Yeshwant R Chillakuru, Youngho Seo, Thienkhai H Vu, and Jae Ho Sohn. Domain specific word embeddings for natural language processing in radiology. *Journal of biomedical informatics*, 113:103665, 2021.

[89] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[90] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[91] Emily Alsentzer, John R Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew McDermott. Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323*, 2019.

[92] Jieh-Sheng Lee and Jieh Hsiang. Patent classification by fine-tuning bert language model. *World Patent Information*, 61:101965, 2020.

[93] Da Yin, Tao Meng, and Kai-Wei Chang. Sentibert: A transferable transformer-based architecture for compositional sentiment semantics. *arXiv preprint arXiv:2005.04114*, 2020.

[94] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[95] Yung-Sung Chuang, Chi-Liang Liu, Hung-Yi Lee, and Lin-shan Lee. Speechbert: An audio-and-text jointly learned language model for end-to-end spoken question answering. *arXiv preprint arXiv:1910.11559*, 2019.

[96] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

[97] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019.

[98] Chris Alberti, Jeffrey Ling, Michael Collins, and David Reitter. Fusion of detected objects in text for visual question answering. *arXiv preprint arXiv:1908.05054*, 2019.

[99] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder

representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.

[100] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019.

[101] Gen Li, Nan Duan, Yuejian Fang, Ming Gong, and Daxin Jiang. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11336–11344, 2020.

[102] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7464–7473, 2019.

[103] Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. Learning video representations using contrastive bidirectional transformer. *arXiv preprint arXiv:1906.05743*, 2019.

[104] Bruno Korbar, Fabio Petroni, Rohit Girdhar, and Lorenzo Torresani. Video understanding as machine translation. *arXiv preprint arXiv:2006.07203*, 2020.

[105] Huaishao Luo, Lei Ji, Botian Shi, Haoyang Huang, Nan Duan, Tianrui Li, Jason Li, Taroon Bharti, and Ming Zhou. Univl: A unified video and language pre-training model for multimodal understanding and generation. *arXiv preprint arXiv:2002.06353*, 2020.

[106] Linjie Li, Yen-Chun Chen, Yu Cheng, Zhe Gan, Licheng Yu, and Jingjing Liu. Hero: Hierarchical encoder for video+ language omni-representation pre-training. *arXiv preprint arXiv:2005.00200*, 2020.

[107] Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, NC Gokul, Avik Bhattacharyya, Mitesh M Khapra, and Pratyush Kumar. Indicnlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained

multilingual language models for indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, 2020.

[108] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de La Clergerie, Djamé Seddah, and Benoît Sagot. Camembert: a tasty french language model. *arXiv preprint arXiv:1911.03894*, 2019.

[109] Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoit Crabbé, Laurent Besacier, and Didier Schwab. Flaubert: Unsupervised language model pre-training for french. *arXiv preprint arXiv:1912.05372*, 2019.

[110] Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. Multilingual is not enough: Bert for finnish. *arXiv preprint arXiv:1912.07076*, 2019.

[111] Wietse De Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. Bertje: A dutch bert model. *arXiv preprint arXiv:1912.09582*, 2019.

[112] P Delobelle, T Winters, and B Berendt. Robbert: A dutch roberta-based language model. arxiv 2020. *arXiv preprint arXiv:2001.06286*.

[113] Wissam Antoun, Fady Baly, and Hazem Hajj. Arabert: Transformer-based model for arabic language understanding. *arXiv preprint arXiv:2003.00104*, 2020.

[114] José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. Spanish pre-trained bert model and evaluation data. *arXiv preprint arXiv:2308.02976*, 2023.

[115] Yuri Kuratov and Mikhail Arkhipov. Adaptation of deep bidirectional multilingual transformers for russian language. *arXiv preprint arXiv:1905.07213*, 2019.

[116] Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. Bertimbau: pretrained bert models for brazilian portuguese. In *Intelligent Systems: 9th Brazilian Conference, BRACIS 2020, Rio Grande, Brazil, October 20–23, 2020, Proceedings, Part I 9*, pages 403–417. Springer, 2020.

[117] Fajri Koto, Afshin Rahimi, Jey Han Lau, and Timothy Baldwin. Indolem and indobert: A benchmark dataset and pre-trained language model for indonesian nlp. *arXiv preprint arXiv:2011.00677*, 2020.

[118] Raphael Scheible, Fabian Thomczyk, Patric Tippmann, Victor Jaravine, and Martin Boeker. Gottbert: a pure german language model. *arXiv preprint arXiv:2012.02110*, 2020.

[119] Raviraj Joshi. L3cube-mahacorpus and mahabert: Marathi monolingual corpus, marathi bert language models, and resources. *arXiv preprint arXiv:2202.01159*, 2022.

[120] Yunfeng Li, Yukun Cao, Qingsheng Zhu, and Zhengyu Zhu. A novel framework for web page classification using two-stage neural network. In *Advanced Data Mining and Applications: First International Conference, ADMA 2005, Wuhan, China, July 22-24, 2005. Proceedings 1*, pages 499–506. Springer, 2005.

[121] Warid Petprasit and Saichon Jaiyen. E-commerce web page classification based on automatic content extraction. In *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 74–77. IEEE, 2015.

[122] Yudong Zhang, Shuihua Wang, Yuxiu Sui, Ming Yang, Bin Liu, Hong Cheng, Junding Sun, Wenjuan Jia, Preetha Phillips, and Juan Manuel Gorriz. Multivariate approach for alzheimer's disease detection using stationary wavelet entropy and predator-prey particle swarm optimization. *Journal of Alzheimer's Disease*, 65(3):855–869, 2018.

[123] GS Tomar, Shekhar Verma, and Ashish Jha. Web page classification using modified na?? ve bayesian approach. In *TENCON 2006-2006 IEEE Region 10 Conference*.

[124] Lay-Ki Soon and Sang Ho Lee. Classifying web pages using information extraction patterns preliminary results and findings. In *2010 Sixth International Conference on Signal-Image Technology and Internet Based Systems*, pages 195–202. IEEE, 2010. doi: 10.1109/SITIS.2010.42.

[125] Chen Shaohong and Wang Zhixing. Web page classification based on semi-supervised naïve bayesian em algorithm. In *2011 IEEE 3rd International Conference on Communication Software and Networks*, pages 242–245. IEEE, 2011.

[126] Hammad Haleem, Pankaj Kumar Sharma, and MM Sufyan Beg. Novel frequent sequential patterns based probabilistic model for effective classification of web documents. In *2014 International Conference on Computer and Communication Technology (ICCCT)*, pages 361–371. IEEE, 2014.

[127] Haijun Zhang, Gang Liu, Tommy WS Chow, and Wenyin Liu. Textual and visual content-based anti-phishing: a bayesian approach. *IEEE transactions on neural networks*, 22(10):1532–1546, 2011.

[128] Weiming Hu, Ou Wu, Zhouyao Chen, Zhouyu Fu, and Steve Maybank. Recognition of pornographic web pages by classifying texts and images. *IEEE transactions on pattern analysis and machine intelligence*, 29(6): 1019–1034, 2007.

[129] Elisabetta Fersini, Enza Messina, and Francesco Archetti. Enhancing web page classification through image-block importance analysis. *Information processing & management*, 44(4):1431–1447, 2008.

[130] E Suganya and Dr S Vijayarani. Web page classification in web mining research-a survey. *International Journal of Innovative Research in Science, Engineering and Technology*, 6:17472–17479, 2017.

[131] Zinah Abdulateef Abbas, Shawkat K. Guirgui, Dr. LaithR. Fleih, and Magda M. Madbouly. Web pages classification using rule-based sys-

tem. 2016. URL https://api.semanticscholar.org/CorpusID:153314942.

[132] Ajay S Patil and BV Pawar. Automated classification of web sites using naive bayesian algorithm. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, pages 519–523. Citeseer, 2012.

[133] Oh-Woog Kwon and Jong-Hyeok Lee. Web page classification based on k-nearest neighbor approach. In *Proceedings of the fifth international workshop on on Information retrieval with Asian languages*, pages 9–15, 2000.

[134] Zhaohui Xu, Fuliang Yan, Jie Qin, and Haifeng Zhu. A web page classification algorithm based on link information. In *2011 10th International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, pages 82–86. IEEE, 2011.

[135] Belmouhcine Abdelbadie and Benkhalifa Mohammed. A clique based web page classification corrective approach. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 2, pages 467–473. IEEE, 2014.

[136] Yi Zheng, Chengcheng Sun, Chengzhang Zhu, Xv Lan, Xiang Fu, and Weihong Han. Lwcs: A large-scale web page classification system based on anchor graph hashing. In *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 90–94. IEEE, 2015.

[137] Abdelbadie Belmouhcine and Mohammed Benkhalifa. Implicit links-based techniques to enrich k-nearest neighbors and naive bayes algorithms for web page classification. In *Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015*, pages 755–766. Springer, 2016.

[138] Oh-Woog Kwon and Jong-Hyeok Lee. Text categorization based on

k-nearest neighbor approach for web site classification. *Information Processing & Management*, 39(1):25–44, 2003.

[139] ESalini Varma P.Kameshwari. Web page classification approach. *SSRG International Journal of Computer Science and Engineering (SSRG-IJCSE*, 4(ue 2–February).

[140] Lijuan Jiao and Liping Feng. Improvement of feature extraction in web page classification. In *2010 2nd International Conference on E-business and Information System Security*, pages 1–3, 2010. doi: 10.1109/EBISS.2010.5473682.

[141] J. Mangai, Dipti Kothari, and Santhosh Vasudevan. A supervised discretization algorithm for web page classification. pages 226–231, 03 2012. ISBN 978-1-4673-1100-7. doi: 10.1109/INNOVATIONS.2012.6207737.

[142] Hamed Jelodar, Seyed Javad Mirabedini, and Ali Harounabadi. Evaluation and analysis of popular decision tree algorithms for annoying advertisement websites classification. In *2015 Fifth International Conference on Communication Systems and Network Technologies*, pages 1025–1029. IEEE, 2015.

[143] Mona Andoohgin Shahri, Mohammad Davarpanah Jazi, Glenn Borchardt, and Mehdi Dadkhah. Detecting hijacked journals by using classification algorithms. *Science and engineering ethics*, 24:655–668, 2018.

[144] Mehdi Dadkhah, Tole Sutikno, Mohammad Davarpanah Jazi, and Deris Stiawan. An introduction to journal phishings and their detection approach. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 13(2):373–380, 2015.

[145] Mehdi Dadkhah and Glenn Borchardt. Guidelines for selecting journals that avoid fraudulent practices in scholarly publishing. *Iranian Journal of Management Studies*, 9:529–538, 2016. URL `https://api.semanticscholar.org/CorpusID:55617172`.

[146] Vicent Estruch, César Ferri, José Hernández-Orallo, and M José Ramírez-Quintana. Web categorisation using distance-based decision trees. *Electronic Notes in Theoretical Computer Science*, 157(2):35–40, 2006.

[147] Hamidreza Alvari, Paulo Shakarian, and Jana Snyder. Semi-supervised learning for detecting human trafficking. *arXiv preprint arXiv:1705.10786*, 2017.

[148] L. Tian, D. Zheng, and C. Zhu. Image classification based on the combination of text features and visual features. *International Journal of Intelligent Systems*, 28(3):242–256, 2013.

[149] Ahmed Abbasi and Hsinchun Chen. Detecting fake websites: The contribution of statistical learning theory. *Computer*, 42(2):56–62, 2009. doi: 10.1109/MC.2009.58.

[150] Elisabetta Fersini, Enza Messina, and Francesca A Pozzi. Improving document classification systems by incorporating image-block evaluation. *Information Processing & Management*, 44(4):1431–1447, 2008.

[151] Adrian Ulges and Armin Stahl. Automatic detection of child pornography using color visual words. In *2011 IEEE International Conference on Multimedia and Expo*, pages 1–6, 2011. doi: 10.1109/ICME.2011.6011977.

[152] Henry Rowley, Yushi Jing, and Shumeet Baluja. Large scale image-based adult-content filtering. pages 290–296, 01 2006.

[153] Aixin Sun, Ee-Peng Lim, and Wee Keong Ng. Web classification using support vector machine. 04 2003. doi: 10.1145/584931.584952. URL `https://doi.org/10.1145/584931.584952`.

[154] Min-Yen Kan. Web page classification without the web page. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*, WWW Alt. '04, page 262–263, New York, NY, USA, 2004. Association for Computing Machinery.

ISBN 1581139128. doi: 10.1145/1013367.1013426. URL https://doi.org/10.1145/1013367.1013426.

[155] Abdelbadie Belmouhcine and Mohammed Benkhalifa. Implicit links based kernel to enrich support vector machine for web page classification. In *2015 10th International Conference on Intelligent Systems: Theories and Applications (SITA)*, pages 1–4, 2015. doi: 10.1109/SITA.2015.7358417.

[156] Abdelbadie Belmouhcine and Mohammed Benkhalifa. Implicit links based web page representation for web page classification. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, WIMS '15, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450332934. doi: 10.1145/2797115.2797125. URL https://doi.org/10.1145/2797115.2797125.

[157] Min Gu, Feng Zhu, Qing Guo, Yanhui Gu, Junsheng Zhou, and Weiguang Qu. Towards effective web page classification. pages 1–2, 11 2016. doi: 10.1109/BESC.2016.7804494.

[158] Rung-Ching Chen and Chung-Hsun Hsieh. Web page classification based on a support vector machine using a weighted vote schema. *Expert Systems with Applications*, 31(2):427–435, 2006. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2005.09.079. URL https://www.sciencedirect.com/science/article/pii/S0957417405002307.

[159] Hamed Jelodar, Yongli Wang, Chi Yuan, and Xiaohui Jiang. A systematic framework to discover pattern for web spam classification. *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 32–39, 2017. URL https://api.semanticscholar.org/CorpusID:7497605.

[160] Wei Wang, Xiao-Dong Lee, An-Lei Hu, and Guang-Gang Geng. Co-training based semi-supervised web spam detection. pages 789–793, 07 2013. ISBN 978-1-4673-5253-6. doi: 10.1109/FSKD.2013.6816301.

[161] Jayanthi K and Dr.Sasikala Subramani. Web link spam identification inspired by artificial immune system and the impact of tpp-fca feature selection on spam classification. *ICTACT Journal on Soft Computing*, 4:633–644, 10 2013. doi: 10.21917/ijsc.2013.0091.

[162] Manuel Egele, Clemens Kolbitsch, and Christian Platzer. Removing web spam links from search engine results. *Journal in Computer Virology*, 7:51–62, 02 2011. doi: 10.1007/s11416-009-0132-6.

[163] Xiaoguang Qi and Brian D Davison. Knowing a web page by the company it keeps. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 228–237, 2006.

[164] Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines. volume 27, pages 807–814, 06 2010.

[165] Mutasem Khalil Alsmadi, Khairuddin Bin Omar, Shahrul Azman Mohd. Noah, and Ibrahim Almarashdah. Performance comparison of multi-layer perceptron (back propagation, delta rule and perceptron) algorithms in neural networks. *2009 IEEE International Advance Computing Conference*, pages 296–299, 2009. URL `https://api.semanticscholar.org/CorpusID:17926363`.

[166] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria E. Presa-Reyes, Mei-Ling Shyu, Shu-Ching Chen, and S. S. Iyengar. A survey on deep learning. *ACM Computing Surveys (CSUR)*, 51: 1 – 36, 2018. URL `https://api.semanticscholar.org/CorpusID:52298127`.

[167] Wei Zhao, Jianbo Ye, Min Yang, Zeyang Lei, Suofei Zhang, and Zhou Zhao. Investigating capsule networks with dynamic routing for text classification. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3110–3119, Brussels, Belgium, oct "-" no 2018. Association for Com-

putational Linguistics. doi: 10.18653/v1/D18-1350. URL https://aclanthology.org/D18-1350.

[168] Libo Qin, Wanxiang Che, Yangming Li, Mingheng Ni, and Ting Liu. Dcr-net: A deep co-interactive relation network for joint dialog act recognition and sentiment classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:8665–8672, 04 2020. doi: 10.1609/aaai.v34i05.6391.

[169] Zhongfen Deng, Hao Peng, Dongxiao He, Jianxin Li, and Philip Yu. Htcinfomax: A global model for hierarchical text classification via information maximization. pages 3259–3265, 01 2021. doi: 10.18653/v1/2021.naacl-main.260.

[170] Danilo Croce, Giuseppe Castellucci, and Roberto Basili. GAN-BERT: Generative adversarial learning for robust text classification with a bunch of labeled examples. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2114–2119, Online, jul 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.191. URL https://aclanthology.org/2020.acl-main.191.

[171] Di Jin, Zhijing Jin, Joey Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:8018–8025, 04 2020. doi: 10.1609/aaai.v34i05.6311.

[172] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7370–7377, Jul. 2019. doi: 10.1609/aaai.v33i01.33017370. URL https://ojs.aaai.org/index.php/AAAI/article/view/4725.

[173] Chen Li, Xutan Peng, Hao Peng, Jianxin Li, and Lihong Wang. Textgtl: Graph-based transductive learning for semi-supervised text

classification via structure-sensitive interpolation. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2680–2686. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/369. URL https://doi.org/10.24963/ijcai.2021/369. Main Track.

[174] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In Chengqing Zong and Michael Strube, editors, *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China, jul 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1162. URL https://aclanthology.org/P15-1162.

[175] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep learning–based text classification: A comprehensive review. *ACM Comput. Surv.*, 54(3), apr 2021. ISSN 0360-0300. doi: 10.1145/3439726. URL https://doi.org/10.1145/3439726.

[176] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *AAAI Conference on Artificial Intelligence*, 2015. URL https://api.semanticscholar.org/CorpusID:16756501.

[177] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, 116:1–20, 12 2016. doi: 10.1007/s11263-015-0823-z.

[178] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015. doi: 10.1038/nature14539.

[179] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998. doi: 10.1109/5.726791.

[180] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In Konstantinos Diamantaras, Wlodek Duch, and Lazaros S. Iliadis, editors, *Artificial Neural Networks – ICANN 2010*, pages 92–101, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15825-4.

[181] Austin Stone, Hua-Yan Wang, Michael Stark, Yi Liu, D. Scott Phoenix, and Dileep George. Teaching compositionality to cnns. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 732–741, 2017. URL `https://api.semanticscholar.org/CorpusID:73264`.

[182] Mara Pistellato, Luca Cosmo, Filippo Bergamasco, Andrea Gasparetto, and Andrea Albarelli. Adaptive albedo compensation for accurate phase-shift coding. pages 2450–2455, 08 2018. doi: 10.1109/ICPR.2018.8545465.

[183] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017. doi: 10.1109/ICEngTechnol.2017.8308186.

[184] Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 562–570, Vancouver, Canada, jul 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1052. URL `https://aclanthology.org/P17-1052`.

[185] Zhuang Chen and Tieyun Qian. Transfer capsule network for aspect

level sentiment classification. pages 547–556, 01 2019. doi: 10.18653/v1/P19-1052.

[186] Liqun Shao, Shun Yao, Xing Zhou, Jia Guo, and Jie Wang. Effective web-page classification using token-string cnn over urls and anchor texts. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, pages 105–111. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2018.

[187] Fabrizio De Fausti, Francesco Pugliese, and Diego Zardetto. Towards automated website classification by deep learning. *arXiv preprint arXiv:1910.09991*, 2019.

[188] Daniel López-Sánchez, Angélica González, and Juan Corchado Rodríguez. Visual content-based web page categorization with deep transfer learning and metric learning. *Neurocomputing*, 338, 04 2019. doi: 10.1016/j.neucom.2018.08.086.

[189] Daniel López-Sánchez, Angélica González, and Juan Corchado Rodríguez. Deep neural networks and transfer learning applied to multimedia web mining. pages 124–131, 06 2018. ISBN 978-3-319-62409-9. doi: 10.1007/978-3-319-62410-5_15.

[190] Amit Kumar Nandanwar and Jaytrilok Choudhary. Web page categorization based on images as multimedia visual feature using deep convolution neural network. *Int. J. Emerg. Technol*, 11:619–625, 2020.

[191] Fudong Nian, Teng Li, Yan Wang, Mingliang Xu, and Jun Wu. Pornographic image detection utilizing deep convolutional neural networks. *Neurocomputing*, 210, 06 2016. doi: 10.1016/j.neucom.2015.09.135.

[192] Xizi Wang, Feng Cheng, Shilin Wang, Huanrong Sun, Gongshen Liu, and Cheng Zhou. Adult image classification by a local-context aware network. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2989–2993, 2018. doi: 10.1109/ICIP.2018.8451366.

[193] Mohamed Moustafa. Applying deep learning to classify pornographic images and videos. *arXiv preprint arXiv:1511.08899*, 2015.

[194] Fawaz Waselallah Alsaade and Mohammed Saeed Alzahrani. Classification and detection of autism spectrum disorder based on deep learning algorithms. *Computational Intelligence and Neuroscience*, 2022, 2022. URL `https://api.semanticscholar.org/CorpusID:247211530`.

[195] C Chinenye Opara, Bo Wei, and Yingke Chen. Htmlphish: Enabling phishing web page detection by applying deep learning techniques on html analysis. *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019. URL `https://api.semanticscholar.org/CorpusID:222097465`.

[196] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[197] Gauri Jain, Manisha Sharma, and Basant Agarwal. Optimizing semantic lstm for spam detection. *International Journal of Information Technology*, 11, 04 2018. doi: 10.1007/s41870-018-0157-5.

[198] George A. Miller. *WordNet: A Lexical Database for English*, volume 38 of *Communications of the ACM*. ACM, New York, NY, USA, November 1995. doi: 10.1145/219717.219748. URL `https://wordnet.princeton.edu/`.

[199] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge, 2018.

[200] Lizhen Tang and Qusay H. Mahmoud. A deep learning-based framework for phishing website detection. *IEEE Access*, 10:1509–1521, 2022. URL `https://api.semanticscholar.org/CorpusID:245449748`.

[201] Kai Tai, Richard Socher, and Christopher Manning. Improved semantic representations from tree-structured long short-term memory networks. 1, 02 2015. doi: 10.3115/v1/P15-1150.

[202] Adji B. Dieng, Chong Wang, Jianfeng Gao, and John William Paisley. Topicrnn: A recurrent neural network with long-range semantic dependency. *ArXiv*, abs/1611.01702, 2016. URL `https://api.semanticscholar.org/CorpusID:6039192`.

[203] Baoxin Wang. Disconnected recurrent neural networks for text categorization. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2311–2320, Melbourne, Australia, jul 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1215. URL `https://aclanthology.org/P18-1215`.

[204] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, 2014. URL `https://api.semanticscholar.org/CorpusID:5590763`.

[205] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.*, 45:2673–2681, 1997. URL `https://api.semanticscholar.org/CorpusID:18375389`.

[206] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *ArXiv*, abs/1802.05365, 2018. URL `https://api.semanticscholar.org/CorpusID:3626819`.

[207] Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip Yu, and Lifang He. A survey on text classification: From traditional to deep learning. *ACM Transactions on Intelligent Systems and Technology*, 13:1–41, 04 2022. doi: 10.1145/3495162.

[208] Waris Quamer, Praphula Kumar Jain, Arpit Rai, Vijayalakshmi Saravanan, Rajendra Pamula, and Chiranjeev Kumar. Sacnn: Self-attentive convolutional neural network model for natural language

inference. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 20 (3), jun 2021. ISSN 2375-4699. doi: 10.1145/3426884. URL `https://doi.org/10.1145/3426884`.

[209] Lahcen Yamoun, Zahia Guessoum, and Christophe Girard. Transformers and attention mechanism for website classification and porn detection. In *European Conference on Advances in Databases and Information Systems*, pages 140–149. Springer, 2023.

[210] Lahcen Yamoun, Zahia Guessoum, and Christophe Girard. Transformer roberta vs. tf-idf for websites content-based classification. In *Deep Learning meets Ontologies and Natural Language Processing, 3rd International Workshop, in conjunction with ESWC 2022*, 2022.

[211] Ferhat Demirkıran, Aykut Çayır, Uğur Ünal, and Hasan Dağ. Website category classification using fine-tuned bert language model. In *2020 5th International Conference on Computer Science and Engineering (UBMK)*, pages 333–336, 2020. doi: 10.1109/UBMK50275.2020.9219384.

[212] Goran Matosevic, Jasminka Doba, and Dunja Mladenić. Using machine learning for web page classification in search engine optimization. *Future Internet*, 13:9, 2021. URL `https://api.semanticscholar.org/CorpusID:231732492`.

[213] Miklós Sebők and Zoltán Kacsuk. The multiclass classification of newspaper articles with machine learning: The hybrid binary snowball approach. *Political Analysis*, 29, 11 2020. doi: 10.1017/pan.2020.27.

[214] C. Raj, A. Agarwal, G. Bharathy, B. Narayan, and M. Prasad. Cyberbullying detection: Hybrid models based on machine learning and natural language processing techniques. *Electronics*, 10(22):2810, 2021. doi: 10.3390/electronics10222810. URL `https://dx.doi.org/10.3390/electronics10222810`.

[215] Selma Ayşe Özel. A genetic algorithm based optimal feature selection for web page classification. In *2011 International Symposium on Inno-*

vations in Intelligent Systems and Applications, pages 282–286, 2011. doi: 10.1109/INISTA.2011.5946076.

[216] Selma Ayşe Özel. A web page classification system based on a genetic algorithm using tagged-terms as features. *Expert Systems with Applications*, 38(4):3407–3415, 2011.

[217] Huaizhi Yan, Xin Zhang, Jiangwei Xie, and Changzhen Hu. *Detecting Malicious URLs Using a Deep Learning Approach Based on Stacked Denoising Autoencoder: 12th Chinese Conference, CTCIS 2018, Wuhan, China, October 18, 2018, Revised Selected Papers*, pages 372–388. 01 2019. ISBN 978-981-13-5912-5. doi: 10.1007/978-981-13-5913-2_23.

[218] Franco Rojas López, Héctor Jiménez-Salazar, and David Pinto. A competitive term selection method for information retrieval. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 468–475. Springer, 2007.

[219] Ganesh Khade, Sudhakar Kumar, and Samit Bhattacharya. Classification of web pages on attractiveness: A supervised learning approach. In *2012 4th International Conference on Intelligent Human Computer Interaction (IHCI)*, pages 1–5. IEEE, 2012.

[220] Leonardo Espinosa-Leal, Anton Akusok, Amaury Lendasse, and Kaj-Mikael Björk. Website classification from webpage renders. In *Proceedings of ELM2019 9*, pages 41–50. Springer, 2021.

[221] Hnin Pwint Myu Wai, Phyu Phyu Tar, and Phyu Thwe. Ontology based web page classification system by using enhanced c4. 5 and naïve bayesian classifiers. In *2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, volume 3, pages 286–291. IEEE, 2018.

[222] Aris Murdiyanto and Muhammad Habibi. Analysis of deep learning approach based on convolution neural network (cnn) for classification of web page title and description text. *Compiler*, 11

(2):51–58, 2022. ISSN 2549-2403. doi: 10.28989/compiler.v11i2.1327. URL https://ejournals.itda.ac.id/index.php/compiler/article/view/1327.

[223] Siti Hawa Apandi, Jamaludin Sallim, Rozlina Mohamed, and Araby Madbouly. Web page classification using convolutional neural network (cnn) towards eliminating internet addiction. In *2021 International Conference on Software Engineering Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM)*, pages 149–154, 2021. doi: 10.1109/ICSECS52883.2021.00034.

[224] *Front Matter*, pages i–xxi. John Wiley Sons, Ltd, 2014. ISBN 9781118914564. doi: https://doi.org/10.1002/9781118914564.

[225] L. Lam and C. Y. Suen. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, 16(9):945–954, 1995.

[226] Paolo Serafini. *Condorcet*, pages 11–20. Springer International Publishing, Cham, 2020. ISBN 978-3-030-38368-8. doi: 10.1007/978-3-030-38368-8_4. URL https://doi.org/10.1007/978-3-030-38368-8_4.

[227] E. M. Albornoz and D. H. Milone. Emotion recognition in never-seen languages using a novel ensemble method with emotion profiles. *IEEE Transactions on Affective Computing*, 8(1):43–53, 2017.

[228] B. Pan, Z. Shi, and X. Xu. Hierarchical guidance filtering-based ensemble classification for hyperspectral images. *IEEE Transactions on Geoscience and Remote Sensing*, PP(99):1–13, 2017.

[229] A. Agrawal and N. K. Mishra. Fusion based emotion recognition system. In *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 727–732, 2016.

[230] L. D. Coronado-De-Alba, A. Rodriguez-Mota, and P. J. E.-Ambrosio. Feature selection and ensemble of classifiers for android malware de-

tection. In *2016 8th IEEE Latin-American Conference on Communications (LATINCOM)*, pages 1–6, 2016.

[231] M. M. Habib, R. A. Welikala, A. Hoppe, C. G. Owen, A. R. Rudnicka, and S. A. Barman. Microaneurysm detection in retinal images using an ensemble classifier. In *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6, 2016.

[232] S. Saha and S. Saha. A novel approach to classify edible oil using multiple classifier fusion based on spectral data. In *2016 International Conference on Intelligent Control Power and Instrumentation (ICICPI)*, pages 108–113, 2016.

[233] R. Toufiq and M. R. Isalm. Face recognition system using soft-output classifier fusion method. In *2016 2nd International Conference on Electrical, Computer Telecommunication Engineering (ICECTE)*, pages 1–4, 2016.

[234] A. Moosavian, M. Khazaee, G. Najafi, M. Kettner, and R. Mamat. Spark plug fault recognition based on sensor fusion and classifier combination using dempster–shafer evidence theory. *Applied Acoustics*, 93: 120–129, 2015.

[235] S. Adhikari and S. Saha. Multiple classifier combination technique for sensor drift compensation using ann knn. In *Advance Computing Conference (IACC), 2014 IEEE International*, pages 1184–1189, 2014.

[236] J. Kremer, K. Steenstrup Pedersen, and C. Igel. Active learning with support vector machines. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4(4):313–326, 2014.

[237] J. Hu and Y. Chen. Offline signature verification using real adaboost classifier combination of pseudo-dynamic features. In *2013 12th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1345–1349, 2013.

[238] L. Junzhao, M. Mohandes, and M. Deriche. A multi-classifier image based vacant parking detection system. In *IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, pages 933–936, Abu Dhabi, UAE, DEC 8-11 2013.

[239] Y.-D. Lan and L. Gao. A new model of combining multiple classifiers based on neural network. In *Emerging Intelligent Data and Web Technologies (EIDWT), 2013 Fourth International Conference on*, pages 154–159, 2013.

[240] A. J. Ma, P. C. Yuen, and J.-H. Lai. Linear dependency modeling for classifier fusion and feature combination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1135–1148, 2013.

[241] M. Mohandes and M. Deriche. Arabic sign language recognition by decisions fusion using dempster-shafer theory of evidence. In *The IEEE Computing, Communications and Applications Conference*, Hong Kong, China, April 1-4 2013.

[242] M. Mohandes and M. Deriche. Arabic sign language recognition by decisions fusion using dempster-shafer theory of evidence. In *Computing, Communications and IT Applications Conference (ComComAp)*, pages 90–94, 2013.

[243] C. Senaras, M. Ozay, and F. T. Yarman Vural. Building detection with decision fusion. 2013.

[244] M. A. A. Siddiqui. Fusion of ecg/eeg for improved automatic seizure detection using dempster shafer theory of evidence. 2013.

[245] U. Ekmekci and Z. Cataltepe. Classifier combination with kernelized eigenclassifiers. In *2013 16th International Conference on Information Fusion (FUSION)*, pages 743–749, 2013.

[246] J. Hou, Z.-S. Feng, and B.-P. Zhang. A graph-theoretic approach to classifier combination. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1017–1020, 2012.

[247] N. Poh and J. Kittler. A unified framework for biometric expert fusion incorporating quality measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):3–18, 2012.

[248] A. Ulaş, O. T. Yıldız, and E. Alpaydın. Eigenclassifiers for combining correlated classifiers. *Information Sciences*, 187:109–120, 2012.

[249] L. Dai and C. Liu. Multiple classifier combination for land cover classification of remote sensing image. In *Information Science and Engineering (ICISE), 2010 2nd International Conference on*, pages 3835–3839, 2010.

[250] H. R. Kalluri, S. Prasad, and L. M. Bruce. Decision-level fusion of spectral reflectance and derivative information for robust hyperspectral land cover classification. *IEEE Transactions on Geoscience and Remote Sensing*, 48(11):4047–4058, 2010.

[251] Y. Zhan, H. Leung, K.-C. Kwak, and H. Yoon. Automated speaker recognition for home service robots using genetic algorithm and dempster–shafer fusion technique. *IEEE Transactions on Instrumentation and Measurement*, 58(9):3058–3068, 2009.

[252] E. Ulaş, M. Semerci, O. T. Yıldız, and E. Alpaydın. Incremental construction of classifier and discriminant ensembles. *Information Sciences*, 179(9):1298–1318, 2009.

[253] M. D. Muhlbaier, A. Topalis, and R. Polikar. Learn. nc: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes. *IEEE Transactions on Neural Networks*, 20(1):152–168, 2009.

[254] S. Tulyakov, S. Jaeger, V. Govindaraju, and D. Doermann. Review of classifier combination methods. In *Machine Learning in Document Analysis and Recognition*, pages 361–386. Springer, 2008.

[255] A. V Bogdanov. Neuroinspired architecture for robust classifier fusion

of multisensor imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 46(5):1467–1487, 2008.

[256] Z. Wu, C.-H. Li, and V. Cheng. Large margin maximum entropy machines for classifier combination. In *Wavelet Analysis and Pattern Recognition, 2008. ICWAPR'08. International Conference on*, volume 1, pages 378–383, 2008.

[257] D.-Q. Han, C.-Z. Han, and Y. Yang. Combination of heterogeneous multiple classifiers based on evidence theory. In *Wavelet Analysis and Pattern Recognition, 2007. ICWAPR'07. International Conference on*, volume 2, pages 573–578, 2007.

[258] B. Quost, T. Denœux, and M.-H. Masson. Pairwise classifier combination using belief functions. *Pattern Recognition Letters*, 28(5):644–653, 2007.

[259] M. Magimai-Doss, D. Hakkani-Tur, O. Cetin, E. Shriberg, J. Fung, and N. Mirghafori. Entropy based classifier combination for sentence segmentation. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–189, 2007.

[260] R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006.

[261] F. Mattern, T. Rohlfing, and J. Denzler. Adaptive performance-based classifier combination for generic object recognition. In *Proc. of International Fall Workshop Vision, Modeling and Visualization (VMV)*, pages 139–146, 2005.

[262] I. Naseem. Combining classifiers using the dempster-shafer theory of evidence. 2005.

[263] G. Fumera and F. Roli. A theoretical and experimental analysis of linear combiners for multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):942–956, 2005.

[264] L. I. Kuncheva. *Combining pattern classifiers: methods and algorithms.* John Wiley Sons, 2004.

[265] J. J. de Oliveira Jr, M. N. Kapp, C. O. de A. Freitas, J. M. de Carvalho, and R. Sabourin. Handwritten recognition with multiple classifiers for restricted lexicon. In *Proceedings. 17th Brazilian Symposium on Computer Graphics and Image Processing*, pages 82–89, 2004.

[266] G. Jain, A. Ginwala, and Y. A. Aslandogan. An approach to text classification using dimensionality reduction and combination of classifiers. In *Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference on*, pages 564–569, 2004.

[267] X. Yi, Z. Kou, and C. Zhang. Classifier combination based on active learning. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 1, pages 184–187, 2004.

[268] S. Gunter and H. Bunke. A new combination scheme for hmm-based classifiers and its application to handwriting recognition. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 2, pages 332–337, 2002.

[269] L. I. Kuncheva. Switching between selection and fusion in combining classifiers: An experiment. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 32(2):146–156, 2002.

[270] J. R. Parker. Combining multiple non-homogeneous classifiers: an empirical approach. In *Cognitive Informatics, IEEE International Conference on*, page 288, 2002.

[271] G. Pasquariello, N. Ancona, P. Blonda, C. Tarantino, G. Satalino, and A. D'Addabbo. Neural network ensemble and support vector machine classifiers for the analysis of remotely sensed data: a comparison. In *Geoscience and Remote Sensing Symposium, 2002. IGARSS'02. 2002 IEEE International*, volume 1, pages 509–511, 2002.

[272] O. Velek, S. Jaeger, and M. Nakagawa. A new warping technique for normalizing likelihood of multiple classifiers and its effectiveness in combined on-line/off-line japanese character recognition. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pages 177–182, 2002.

[273] W. Wang, A. Brakensiek, and G. Rigoll. Combination of multiple classifiers for handwritten word recognition. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pages 117–122, 2002.

[274] T. D. Pham. Combination of multiple classifiers using adaptive fuzzy integral. In *Artificial Intelligence Systems, 2002.(ICAIS 2002). 2002 IEEE International Conference on*, pages 50–55, 2002.

[275] R. P. W. Duin. The combining classifier: to train or not to train? In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 2, pages 765–770, 2002.

[276] R. P. W. Duin and D. M. J. Tax. Experiments with classifier combining rules. In *Multiple Classifier Systems*, pages 16–29. Springer, 2000.

[277] A. Rahman and M. Fairhurst. Decision combination of multiple classifiers for pattern classification: Hybridisation of majority voting and divide and conquer techniques. In *Applications of Computer Vision, 2000, Fifth IEEE Workshop on*, pages 58–63, 2000.

[278] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[279] D. J. Miller and L. Yan. Ensemble classification by critic-driven combining. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 2, pages 1029–1032, 1999.

[280] H. Schwenk. Using boosting to improve a hybrid hmm/neural network speech recognizer. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1009–1012, 1999.

[281] L. I. Kuncheva, J. C. Bezdek, and M. A. Sutton. On combining multiple classifiers by fuzzy templates. In *Fuzzy Information Processing Society-NAFIPS, 1998 Conference of the North American*, pages 193–197, 1998.

[282] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[283] J. Kittler, M. Hatef, and R. P. W. Duins. Combining classifiers. In *IEEE Proceedings of ICPR 96*, pages 897–901, 1996.

[284] T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, 1994.

[285] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):418–435, 1992.

[286] A. El Khatib, N. Werghi, and H. Al-Ahmad. Enhancing automatic polyp detection accuracy using fusion techniques. In *IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1–4, 2016.

[287] J. Kittler. Combining classifiers: A theoretical framework. *Pattern Analysis and Applications*, 1(1):18–27, 1998.

[288] M. Mohandes, S. Aliyu, and M. Deriche. Arabic sign language recognition using the leap motion controller. In *Industrial Electronics (ISIE), 2014 IEEE 23rd International Symposium on*, pages 960–965, 2014.

[289] M. Liu, K. Li, and R. Zhao. A boundary based classifier combination method. In *2009 Chinese Control and Decision Conference (CCDC)*, pages 3777–3782, 2009.

[290] Xiaoguang Qi and Brian D Davison. Web page classification: Features and algorithms. *ACM computing surveys (CSUR)*, 41(2):1–31, 2009.

[291] Abhimanyu Panwar, Iosif-Viorel Onut, and James Miller. Towards real time contextual advertising. In *International Conference on Web Information Systems Engineering*, pages 445–459. Springer, 2014.

[292] Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. Intelligent rule-based phishing websites classification. *IET Information Security*, 8(3):153–160, 2014.

[293] Jitendra Kumar, A. Santhanavijayan, B. Janet, Balaji Rajendran, and B.S. Bindhumadhava. Phishing website classification and detection using machine learning. In *2020 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6, 2020. doi: 10.1109/ICCCI48352.2020.9104161.

[294] Shafaizal Shabudin, Nor Samsiah Sani, Khairul Akram Zainal Ariffin, and Mohd Aliff. Feature selection for phishing website classification. *International Journal of Advanced Computer Science and Applications*, 11(4), 2020.

[295] Shaobo Zhong and Dongsheng Zou. Web page classification using an ensemble of support vector machine classifiers. *Journal of Networks*, 6 (11):1625, 2011.

[296] M Janaki Meena, KR Chandran, A Karthik, and A Vijay Samuel. A parallel aco algorithm to select terms to categorise longer documents. *International Journal of Computational Science and Engineering*, 6 (4):238–248, 2011.

[297] M Janaki Meena, KR Chandran, A Karthik, and A Vijay Samuel. An enhanced aco algorithm to select features for text categorization and

its parallelization. *Expert Systems with Applications*, 39(5):5861–5871, 2012.

[298] Vladimír Bartík. Text-based web page classification with use of visual information. In *2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 416–420. IEEE, 2010.

[299] Sonal D Vaghela and Pinal Patel. Web page classification techniques-a comprehensive survey 1. 2014.

[300] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct): 2825–2830, 2011.

[301] Zeyi Wen, Jiashuai Shi, Qinbin Li, Bingsheng He, and Jian Chen. ThunderSVM: A fast SVM library on GPUs and CPUs. *Journal of Machine Learning Research*, 19:797–801, 2018.

[302] Radim Rehurek and Petr Sojka. Gensim–python framework for vector space modelling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2), 2011.

[303] Bhargav Srinivasa-Desikan. *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd, 2018.

[304] Ye Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820, 2015. URL `http://arxiv.org/abs/1510.03820`.

[305] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank

Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-lib` `pdf`.

[306] Winda Sari, Dian Rini, and Reza Malik. Text classification using long short-term memory with glove features. *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, 5(2):85–100, 2019. ISSN 2338-3062. URL `http://journal.uad.ac.id/index.php/JITEKI/article/view/15021`.

[307] Arun S. Maiya. ktrain: A low-code library for augmented machine learning. *arXiv preprint arXiv:2004.10703*, 2020.

[308] Yurii Nesterov. *A Method of Solving a Convex Programming Problem with Convergence Rate O(1/k²)*, volume 27. Springer, 1983.

[309] James Bergstra, Daniel Yamins, and David D. Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. 2013.

[310] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[311] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[312] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. 2020.

[313] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, and Quoc V Le. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, 2019.

[314] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations (ICLR)*, 2020.

[315] Zihao Ye, Qipeng Guo, Quan Gan, Xipeng Qiu, and Zheng Zhang. Bp-transformer: Modelling long-range context via binary partitioning. *arXiv preprint arXiv:1911.04070*, 2019.

[316] Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding, 2019.

[317] Sinong Wang, Zeming Li, Madian Khabsa, Hanqing Fang, Hao Ma, and Jiliang Tang. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[318] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Adaptive attention span in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2911–2922, 2020.

[319] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *arXiv preprint arXiv:2003.05997*, 2020.

[320] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, 2020.

[321] Krzysztof Choromanski, Vitaly Likhosherstov, David Dohan, Xingyou

Song, Alain Gane, Tamas Sarlos, ..., and Eugene Belilovsky. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[322] Mohit Tuteja and Daniel González Juclà. Long text classification using transformers with paragraph selection strategies. In Daniel Preoțiuc-Pietro, Catalina Goanta, Ilias Chalkidis, Leslie Barrett, Gerasimos (Jerry) Spanakis, and Nikolaos Aletras, editors, *Proceedings of the Natural Legal Language Processing Workshop 2023*, pages 17–24, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.nllp-1.3. URL `https://aclanthology.org/2023.nllp-1.3`.

[323] Aman Jaiswal and Evangelos E. Milios. Breaking the token barrier: Chunking and convolution for efficient long text classification with bert. *ArXiv*, abs/2310.20558, 2023. URL `https://api.semanticscholar.org/CorpusID:264818633`.

[324] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Ijcai*, 14:1137–1145, 1995.

[325] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning*. Springer series in statistics New York, 2001.

[326] George Forman and Martin Scholz. Apples-to-apples in cross-validation studies: Pitfalls in classifier performance measurement. *ACM SIGKDD Explorations Newsletter*, 12(1):49–57, 2010.

[327] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. Data mining: Practical machine learning tools and techniques. In *Morgan Kaufmann*, 2016.

[328] William G Cochran. The comparison of percentages in matched samples. *Biometrika*, 37(3/4):256–266, 1950.

[329] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[330] Chang Liu, Xueying Chen, and Wayne Xin Zhao. Efficient adaptation of pretrained transformers for text classification using quantized low-rank matrices. *arXiv preprint arXiv:2110.05201*, 2021.

[331] Jonas Pfeiffer, Abhishek Kamath, Andreas Rücklé, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, 2020.

[332] Mohamed Zohir Koufi, Zahia Guessoum, Amor Keziou, and Itheri Yahiaoui. Toward website classification. In *2023 IEEE International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pages 306–310. IEEE, 2023.

[333] Mohamed Zohir Koufi, Zahia Guessoum, Amor Keziou, and Itheri Yahiaoui. Text chunking to improve website classification. In *International Conference on Optimization, Learning Algorithms and Applications*, pages 197–216. Springer, 2024.