Ph.D Dissertation

Presented by :

Maroua LOUAIL

Dimensionality Reduction in Machine Learning for Arabic Text Classification

Abstract

Text classification is the automated process of assigning predefined labels or categories to text based on its content. This process helps organize vast amounts of textual data, simplifies management, enables efficient searches, and extracts valuable knowledge. The computational analysis of the Arabic language plays a crucial role in addressing its growing global significance. As the fourth most widely used language online, Arabic has driven the emergence of Arabic Text Classification (ATC) as a key research area. However, the field of ATC faces considerable challenges, primarily due to the linguistic complexity of the language and the high computational demands of its processing, which can impact the performance of real-time systems. This dissertation aims to bridge the gap between effectiveness and efficiency in ATC, particularly in resource-constrained environments.

The first objective of this research is to review existing ATC techniques, including preprocessing methods, vectorization strategies, dimensionality reduction techniques, and both classical machine learning and deep learning models, in order to provide a comprehensive understanding of current approaches. The second objective is to propose three innovative methods to enhance computational efficiency through dimensionality reduction while improving or at least maintaining high classification effectiveness. These methods are specifically designed for Modern Standard Arabic (MSA) text classification and are evaluated against state-of-the-art methods.

The dissertation presents the use of Principal Component Analysis (PCA), Distance-Based Meta-Features (DBMFs) for feature extraction, and the development of a new hybrid approach called "*Tasneef*", which addresses computational challenges in Arabic text processing and outperforms state-of-the-art deep learning models and dimensionality reduction techniques. Through these contributions, this dissertation advances the state of the art in ATC by focusing on dimensionality reduction, which improves classification accuracy and reduces memory usage and runtime.

Keywords: Natural language processing, Arabic text classification, Dimensionality reduction, feature extraction, Meta-features, Word embeddings.

ملخص

تصنيف النصوص هو عملية آلية تهدف إلى تعيين تسميات أو فئات محددة مسبقًا للنص بناءً على محتواه. تساعد هذه العملية في تنظيم كميات هائلة من البيانات النصية، وتبسط إدارتها، وتمكّن من عمليات بحث فعالة، كما تساهم في استخراج المعرفة القيمة. تلعب التحليلات الحسابية للغة العربية دورًا حاسمًا في التعامل مع أهميتها العالمية المتزايدة. وبما أن العربية هي رابع أكثر لغة مستخدمة على الإنترنت، فقد برز تصنيف النصوص العربية كأحد مجالات البحث الرئيسية. ومع ذلك، يواجه مجال تصنيف النصوص العربية تحديات كبيرة، ويعود ذلك أساسًا إلى التعقيد اللغوي للغة والطلب العالي على الموارد الحسابية في معالجتها، مما يمكن أن يؤثر على أداء أنظمة الزمن الحقيقي. تهدف هذه الرسالة إلى سد الفجوة بين الفعالية والكفاءة في تصنيف النصوص العربية، خصوصًا في البيئات ذات الموارد المحدودة.

الهدف الأول من هذا البحث هو استعراض تقنيات تصنيف النصوص العربية الحالية، بما في ذلك طرق المعالجة المسبقة، استراتيجيات التمثيل، تقنيات تقليل الأبعاد، ونماذج التعلم الآلي التقليدية والعميقة، من أجل تقديم فهم شامل للنهج الحالية. الهدف الثاني هو اقتراح ثلاث طرق مبتكرة تركز على تعزيز الكفاءة الحسابية من خلال تقليل الأبعاد، مع تحسين أو على الأقل الحفاظ على فعالية تصنيف عالية. تم تصميم هذه الطرق خصيصًا لتصنيف نصوص اللغة العربية الفصحى وتقييمها مقابل الطرق المتطورة.

تستعرض الرسالة استخدام تحليل المكونات الرئيسية ، والميزات الفوقية المعتمدة على المسافة لاستخراج السمات، وتطوير نهج هجين جديد يسمى «تصنيف»، والذي يعالج التحديات الحسابية في معالجة النصوص العربية ويتفوق على النماذج الحديثة للتعلم العميق وتقنيات تقليل الأبعاد. من خلال هذه المساهمات، تُحرز هذه الرسالة تقدمًا في تصنيف النصوص العربية من خلال التركيز على تقليص الأبعاد مما يحسن من دقة التصنيف و يقلل من استهلاك الذاكرة ووقت التنفيذ.

الكلمات المفتاحية: معالجة اللغة الطبيعية، تصنيف النصوص العربية، تقليص الأبعاد، استخراج السمات، الميزات الفوقية، تمثيلات الكلمات.

Résumé

La classification de texte est le processus automatisé d'attribution d'étiquettes ou de catégories prédéfinies à un texte en fonction de son contenu. Ce processus aide à organiser de vastes quantités de données textuelles, simplifie leur gestion, permet des recherches efficaces et extrait des connaissances précieuses. L'analyse computationnelle de la langue arabe joue un rôle crucial dans la prise en compte de son importance croissante à l'échelle mondiale. En tant que quatrième langue la plus utilisée en ligne, l'arabe a favorisé l'émergence de la classification de texte arabe en tant que domaine de recherche clé, malgré les défis substantiels liés à sa complexité linguistique et aux besoins computationnels intenses nécessairesà son traitement, lesquels compromettent souvent les performances des systèmes en temps réel. Cette thèse se propose de pallier ces limitations en optimisant à la fois l'efficacité et la performance de la classification, notamment dans des environnements à ressources limitées.

Le premier objectif de cette recherche consiste en une analyse critique et systématique des méthodes actuelles, couvrant les techniques de prétraitement, les stratégies de vectorisation, les approches de réduction de dimensionnalité, ainsi que les modèles d'apprentissage automatique et d'apprentissage profond. Ce travail de synthèse vise à dresser un état de l'art détaillé, indispensable pour une compréhension approfondie des approches existantes. Le second objectif est l'introduction de trois méthodologies inédites axées sur l'amélioration de l'efficacité computationnelle par la réduction de dimensionnalité, tout en améliorant ou en maintenant au moins une haute efficacité de classification. Conçues spécifiquement pour la classification de texte en arabe standard, ces méthodologies sont rigoureusement évaluées en comparaison avec les techniques de pointe actuelles.

Les contributions de cette thèse incluent l'application de l'analyse par composantes principales (ACP) et l'utilisation des méta-caractéristiques basées sur les distances pour l'extraction de caractéristiques, ainsi que le développement d'une nouvelle approche hybride, nommée "*Tasneef*". Cette dernière, spécialement conçue pour surmonter les défis computationnels inhérents au traitement du texte en arabe, démontre une supériorité par rapport aux modèles d'apprentissage profond et aux techniques de réduction de dimensionnalité les plus avancés. Grâce à ces contributions, cette thèse fait progresser l'état de l'art dans la classification de texte arabe en se concentrant sur la réduction de dimensionnalité, ce qui améliore la précision de la classification, réduit l'utilisation de la mémoire et diminue le temps d'exécution.
Mots-clés: Traitement automatique du langage naturel, Classification des textes arabes, Réduction de dimension, Extraction de caractéristiques, Méta-caractéristiques, Représentations vectorielles de mots.

Declaration of Authorship

I, MAROUA LOUAIL, declare that this thesis, titled "Dimensionality Reduction in Machine Learning for Arabic Text Classification," is entirely my original work. I certify that:

- The research was conducted primarily during my time as a candidate for a degree at this University.
- Published works consulted are properly credited.
- All quotations are accurately sourced, and apart from these, the work is exclusively mine.
- Significant sources of support are duly acknowledged.

Signed: Maroua LOUAIL

Date: 16/11/2024

وَالْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ

[الصافات الآية :182]

Acknowledgements

I would like to express my heartfelt gratitude to **God** for His unwavering guidance and strength throughout my PhD journey.

My deepest gratitude go to my advisor, **Pr. Chafia Kara-Mohamed**, whose steadfast support and invaluable guidance have been crucial to the success of my research. Your encouragement has been a pillar of my achievements.

I am also profoundly grateful to **Pr. Aboubekeur Hamdi-Cherif** for your insightful advice and significant contributions, which have greatly enriched my work.

I would like to extend my sincere gratitude to the panel of examiners for their time, insightful feedback, and invaluable contributions to this work. Their expertise and constructive critiques have significantly enriched this research, helping to shape it into its final form. I am deeply appreciative of their guidance and support throughout the examination process.

A special thanks goes to my parents for their boundless love and support throughout these years. Your belief in me has been the cornerstone of my success.

Lastly, I want to express my deep appreciation to my sister, brother, nephew, and niece. Your constant support and understanding have been a source of immense joy and motivation.

Contents

List of Figures

List of Tables

List of Abbreviations

AIR	**A**ccuracy **I**mprovement **R**atio
ATC	**A**rabic **T**exe **C**lassification
BERT	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
BiRNN	**B**idirectional **R**ecurrent **N**eural **N**etwork
BoW	**B**ag **O**f **W**ords
CBOW	**C**ontinuous **B**ag**O**f **W**ords
CFS	**C**orrelation **F**eature **S**election
CNN	**C**onvolutional **N**eural **N**etwork
CEV	**C**umulative **E**xplained **V**ariance
DBMFs	**D**istance-**B**ased **M**eta-**F**eatures
DT	**D**ecision **T**rees
GloVe	**G**lobal **V**ectors for Word Representation
GPU	**G**ated **R**ecurrent **U**nit
IDF	**I**nverse **D**ocument **F**requency
IG	**I**nformation **G**ain
LDA	**L**inear **D**iscriminant **A**nalysis
LSA	**L**atent **S**emantic **A**nalysis
LSI	**L**atent **S**emantic **I**ndexing
LSTM	**L**ong **S**hort-**T**erm **M**emory
ML	**M**achine **L**earning
MI	**M**utual **I**nformation
MSA	**M**odern **S**tandard **A**rabic
MLP	**M**ulti-**L**aye **P**erceptron
NLP	**N**atural **L**anguage **P**rocessing
NER	**N**amed **E**ntity **R**ecognition
NB	**N**aive **B**ayes
NMF	**N**on-Negative **M**atrix **F**actorization
OOV	**O**ut-**O**f-**V**ocabulary
PCA	**P**rincipal **C**omponent **A**nalysis
PCs	**P**rincipal **C**omponents

RF	**R**andom **F**orest
SOTA	**S**tate **O**f **T**he **A**rt
SVD	**S**ingular **V**alue **D**ecomposition
SVM	**S**upport **V**ector **M**achine
TC	**T**ext **C**lassification
TF	**T**erm **F**requency
TF-IDF	**T**erm **F**requency-**I**nverse **D**ocument **F**requency
Word2Vec	**W**ord to **V**ector
kNN	k-**N**earest **N**eighbor
GPT	**G**enerative **P**re-trained **T**ransformer
w.r.t	**w**ith **r**espect **t**o
XGB	**E**xtreme **G**radient **B**oosting

Introduction

The computational analysis of the Arabic language is critically important given its significant global presence. Arabic ranks as the fourth most widely used language online, following English, Chinese, and Spanish, with Arab users comprising 5.2% of the global internet population and being the fastest-growing language over the past five years. Notably, 53% of the Arab population has internet access[1]. Arabic is the official language of all Arab League member states and a primary language in most countries of the Organization of Islamic Cooperation (OIC), making it a key global language. It serves as a gateway to the rich cultural and historical heritage of the Arab and Islamic world, spanning over fourteen centuries. A grasp of the language is essential for apprehending regional perspectives on geopolitical events and for effective communication within the Arab world. Moreover, Arabic-speaking nations assume important roles in global affairs owing to their geopolitical, economic, and strategic position, particularly as key contributors to the global energy market. As such, the Arabic language becomes crucial for diplomatic and economic relations, negotiation of trade agreements, and for understanding economic local policies.

Arabic is currently widely used in regional media, including news outlets, television, and online platforms. Additionally, Arabic-speaking communities are dispersed globally due to migration and diaspora, making the language relevant in multicultural societies outside the Arab world, influencing social dynamics, politics, and international relations (Redkin and Bernikova, 2016). These arguments underscore the importance of Arabic language computational processing, especially given the significant time investment required to master it. Among the various computational tasks, Arabic text classification (ATC) emerges as one of the most critical areas of research. ATC has broad applications across various domains. In social media, it aids content management, enhances safety by supporting incident monitoring, and detects cyber threats on platforms like Twitter and Instagram. It also helps in author profiling and sentiment analysis. In healthcare, ATC categorizes health-related information, revealing that half of medical tweets from professionals are false (Alnemer et al., 2015). In the legal sector, ATC classifies Arabic legal documents to improve access and data protection. It is also applied in recommendation systems and *fatwā* classification in socio-religious contexts. ATC is widely used for topical classification in news, as well as in tasks like dialect identification and web page clustering.

The major challenge faced by any text classification system is the "Curse of Dimensionality". This term introduced in (Bellman, 1957), describes the challenges of analyzing high-dimensional data. Text documents, with numerous unique words as features, often lead

[1] www.internetworldstats.com/stats7.htm

to high-dimensional data, causing increased storage needs, higher computational complexity, and a higher risk of overfitting. Dimensionality reduction techniques, including feature selection and extraction, help mitigate these challenges by reducing the number of features, improving computational efficiency, enhancing model accuracy, and making complex data more interpretable and easier to visualize. The amount of research conducted on ATC is relatively limited compared to the extensive work done on English text classification. One of the main reasons for this gap is the significant challenges ATC faces, including linguistic complexity and limited resources. Recently, deep learning techniques have shown promising results in ATC (Elnagar, Al-Debsi, and Einea, 2020), though they often involve high computational costs, particularly in terms of runtime, which can impact the performance of real-time, stream-based applications. Additionally, achieving accurate and robust ATC with minimal computational resources is essential, especially for mid-sized companies or research groups with limited resources. Most research in the ATC field prioritizes effectiveness, often overlooking efficiency considerations. To address these challenges, this doctoral dissertation sets a two-fold objective: Firstly, we delve into ATC-related works by identifying the strengths and limitations of datasets, preprocessing methods, vectorization, dimensionality reduction techniques, as well as classical machine learning and deep learning models commonly used in the field, thereby providing a comprehensive understanding of the research landscape. Secondly, based on the conclusions drawn from ATC-related works , we propose three approaches for ATC, with a particular focus on Arabic topical classification. The proposed approaches aim to address the computational challenges (memory consumption and runtime) associated with ATC through dimensionality reduction methods, while maintaining a high level of effectiveness. These approaches will be compared with state-of-the-art methods in ATC to evaluate their performance.

Dissertation outline

The dissertation is organized into four chapters. The first chapter provides a comprehensive overview of foundational concepts and prior research related to Arabic document classification, with a specific emphasis on dimensionality reduction techniques. The remaining three chapters present the main contributions to Arabic Text Classification (ATC) using dimensionality reduction. These chapters are primarily based on our published papers during the PhD period, with minor modifications and removal of overlapping content where necessary.

Chapter 1: This chapter thoroughly examines the core concepts and existing research on

Arabic document classification, covering text classification basics, Arabic linguistic complexities, vectorization, dimensionality reduction, machine learning, and deep learning approaches. It also reviews datasets, preprocessing, vectorization, dimensionality reduction, machine learning, deep learning, and evaluation methods used in Arabic topical text classification, laying the groundwork for the following chapters.

Chapter 2: This chapter presents our first contribution, in which we proposed the use of Principal Component Analysis (PCA) as a feature extraction method for Arabic text classification. We assessed its impact on both the effectiveness and efficiency of several prominent classifiers, including support vector machines, random forests, decision trees, k-nearest neighbors, and logistic regression. By incorporating PCA, we aimed to address the challenges posed by high-dimensional data in Arabic text processing, thereby improving both computational efficiency and model performance. To the best of our knowledge, this was the first application of PCA to Arabic text classification. This chapter is based on our conference paper, originally presented in (Louail, Kara-Mohamed Hamdi-Cherif, and Hamdi-Cherif, 2021)

Chapter 3: This chapter presents our second contribution, where we introduced an additional preprocessing step to the classification pipeline by generating distance-based meta-features derived from the original TF-IDF representations. Specifically, we focus on four types of distance-based meta-features: CosKNN, L2KNN, CosCent, and L2Cent, and evaluate their effectiveness and efficiency as dimensionality reduction techniques for ATC. The impact of these features is assessed using four widely used classifiers: k-Nearest Neighbors, Logistic Regression, Random Forest, and Support Vector Machine. This work highlights the significance of emphasizing the pre-processing phase over the classifier algorithm to achieve better text classification performance while minimizing time-related costs. To the best of our knowledge, this represents the first application of distance-based meta-features in the context of Arabic text classification. This chapter is based on our research publication (Louail and Kara-Mohamed, 2023)

Chapter 4: In this chapter, we proposed *Tasneef*, a novel hybrid approach to tackle computational challenges by reducing memory usage and runtime overhead for actual Arabic text classification.The main contributions of chapter can be described as follows:

- Design DBMFs that combine both local information, achieved through k-nearest neighbor (kNN) DBMFs, and global information through centroid DBMFs, as they have been demonstrated to yield significant results when applied to classify English text (Canuto et al., 2018).

- Develop a hybrid representation through the integration of DBMFs with pretrained fastText embeddings integrating both statistical and semantic properties.
- Emphasize the importance of memory consumption and runtime as crucial factors for analysis, focusing on the performance gains achieved by *Tasneef* while maintaining classification effectiveness without significant losses. These conflictual aspects, taken concurrently, have been largely overlooked in existing ATC literature (Wahdan, Al-Emran, and Shaalan, 2023)
- Undertake two series of experiments: one for choosing the best DBMFs, the other for comparing *Tasneef* with state-of-the-art (SOTA) methods, including advanced deep-learning models and feature reduction methods.

This work was published as an academic article in (Louail, Kara-Mohamed, and Hamdi-Cherif, 2024)

Chapter 1

Background and Related Works

1.1 Introduction

This chapter provides a detailed overview of the foundational concepts and prior research related to Arabic document classification, with a particular focus on the use of dimensionality reduction techniques. The chapter begins with an exploration of text classification fundamentals, followed by a detailed discussion of the unique properties of the Arabic language that make classification tasks more complex. Various text vectorization techniques are then reviewed, including classical methods like one-hot encoding, bag-of-words, and more advanced approaches such as word embeddings. Additionally, dimensionality reduction techniques, crucial for handling high-dimensional data in text classification, are examined. The chapter also covers both classical machine learning and deep learning approaches, providing insights into how each is applied to Arabic text classification. Finally, the chapter delves into related works, discussing the datasets, preprocessing techniques, and models commonly employed in the field, offering a contextual understanding of the research landscape.

1.2 Text Classification : Key Concepts and General Pipeline

1.2.1 Text Classification Levels

Text classification can be broadly categorized into four levels (Kowsari et al., 2019), depending on the granularity of the classification task:

1. **Document level:** At the document level, the entire document is treated as a single unit and labeled with the relevant class or classes. This is useful for tasks like topic categorization.

2. **Paragraph level:** At the paragraph level, a portion of the document is considered as a unit and labeled with the relevant class or classes. This is useful for more detailed categorization where different paragraphs might belong to different categories.

3. **Sentence level:** At the sentence level, a single sentence (a portion of a paragraph) is assigned the relevant class or classes. This can be used for tasks like identifying sentence-level sentiment or extracting specific information from text.

4. **Sub-Sentence level:** At the sub-sentence level, a portion of a sentence , such as phrases or clauses is labeled with the relevant class or classes. This level of granularity is useful for tasks like named entity recognition (NER) or aspect-based sentiment analysis.

1.2.2 Types of Text Classification

Text classification is a diverse field with various approaches tailored to different tasks. Some commonly discussed types of text classification in the academic literature include:

- **Single-Label Classification:**This type involves categorizing texts into one of several possible classes, where each text is assigned a single label. This can be further divided into two classes:

  1. **Binary Classification:** A specific case of single-label classification where texts are categorized into one of two possible classes. For example, in email spam detection, emails are classified as either "spam" or "not spam" (Raj et al., 2018).
  2. **Multi-Class Classification:** Another form of single-label classification where texts are categorized into one of several distinct classes. An example is the categorization of news articles into categories such as "politics," "sports," or "technology" (Einea, Elnagar, and Debsi, 2019).

- **Multi-Label Classification:** In this approach, multiple labels are assigned to a single text (Elnagar, Al-Debsi, and Einea, 2020). For instance, a news article might be tagged with multiple labels like "politics," "economy," and "international" . This method is beneficial when a text can belong to more than one category simultaneously.

- **Hierarchical Classification:** This classification method organizes texts into categories with a hierarchical structure. For example, a document might first be classified into a broad category such as "Science" and then into more specific subcategories like "Physics" or "Chemistry" (Silla and Freitas, 2010). Hierarchical classification is useful for tasks requiring a structured categorization scheme.

1.2.3 Arabic Text Classification General Pipeline

Figure 1.1 illustrates the pipeline for Arabic text classification task using both Classical Machine Learning and Deep Learning approaches, highlighting the distinct methodologies employed by each.

1. **Preprocessing:** Both approaches start with preprocessing the raw text, a crucial step to clean and prepare the data for further analysis. This process generally includes:

   - **Tokenization:** Tokenization involves dividing a text into smaller linguistic components, known as tokens. These tokens can be words, subwords, or characters, depending on the needs of the task and the specific requirements of the analysis.
   - **Text cleaning:** This step includes eliminating digits, special characters, non-Arabic words, and Arabic stop-words (e.g., "و" (and), "في" (in), etc.) that might not contribute to the classification task.
   - **Normalization:** Normalization involves standardizing text data by reducing variation and ensuring uniformity. For Arabic text, this typically means replacing different forms of alif (أ, إ, آ) with a standard alif (ا), converting yā (ي) to alif maksoorah (ى), and changing tā marbootah (ة) to hā (ه).
   - **Stemming/Lemmatization:** Generally, one method is used, either stemming or lemmatization. Both methods are used to reduce words to their base or root forms, but they differ in approach and accuracy. Stemming removes affixes based on heuristic rules to produce a "stem," which may not always be a valid word. It is faster but less precise and can result in incorrect forms. Lemmatization, in contrast, reduces words to their canonical form, or "lemma," by considering context and using dictionary lookups and morphological analysis. This makes lemmatization more accurate but also more computationally intensive and resource-dependent.

2. **Classical Machine Learning Approach:** In the classical machine learning approach to text classification, the process typically involves two key phases: feature extraction or feature selection, and classification.

   (a) **Feature Extraction \Feature Selection:** Feature Extraction and Feature Selection are both techniques used in text processing and machine learning. Feature Extraction involves creating a new set of features from the original features, which often results in a more compact representation with improved quality. Conversely, Feature Selection entails choosing a subset from the existing features, concentrating on preserving those that are most pertinent and eliminating those that are less useful or redundant. While both techniques work towards reducing the dimensionality of the data, they do so through different processes. Feature extraction methods, covering vectorization and transformation techniques, are discussed in Section 1.4 and Subsection 1.5.1. Feature selection methods are reviewed in Subsection 1.5.2.

(b) **Classifier:** Once the raw text has been preprocessed and the most relevant features for the classification task have been extracted or selected, these features are input into a machine learning classifier that has been trained on labeled data. By learning from patterns and relationships within the training data, the classifier can make informed predictions on new, unseen examples. This approach ensures that the classifier utilizes the most relevant information, enhancing both its performance and accuracy. The classifiers employed for text classification task are thoroughly described in Section 1.6

3. **Deep Learning Approach:** In this approach, neural network models like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) handle both feature extraction and classification in a unified process. These models automatically learn to discover and extract important features from text data through their layered architecture, eliminating the need for manual feature engineering. This end-to-end approach enables the models to capture intricate patterns and relationships within the text, enhancing the effectiveness and accuracy of text classification tasks. Deep learning-based approach techniques employed for text classification are thoroughly examined in Section 1.7

4. **Evaluation:** Experimental evaluation is essential for assessing a classifier's ability to generalize to new text and for optimizing the model during training. Key performance metrics for text classifiers include accuracy, precision, recall, and the F1-score.

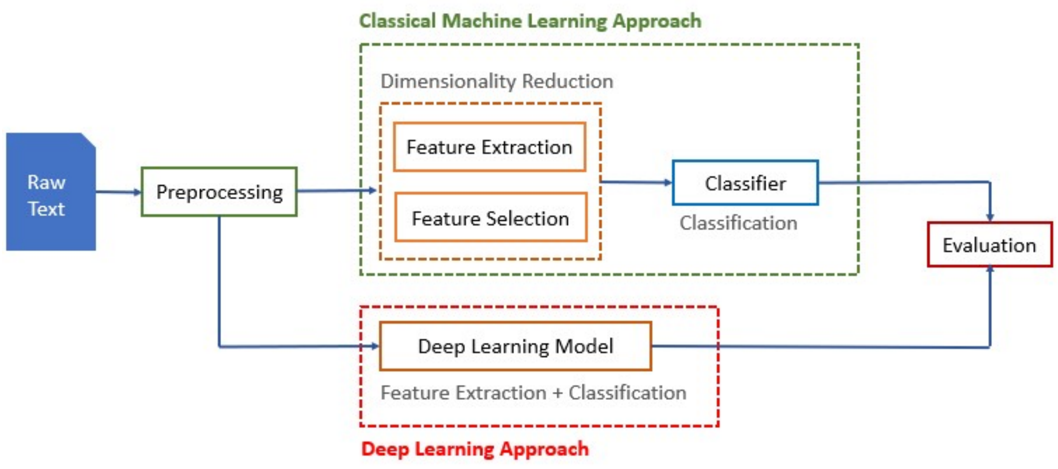


FIGURE 1.1: Text classification general pipeline .

1.3 Arabic Language Properties and TC Challenges

1.3.1 Importance of the Arabic Language

The Arabic language is one of the six official languages of the United Nations[1], alongside Chinese, English, French, Russian, and Spanish. Geographically, Arabic is an Afro-Asiatic language and is the most widely spoken within the Semitic branch of the Afro-Asiatic language family. It is used by over 473.27 million people [2] across 22 countries in the Arab world. Additionally, Arabic is considered a sacred language by approximately 2.03 billion Muslims worldwide[3], which represents over 25% of the world's total population of 8.12 billion, as the *Qur'an* and *Hadith*—central texts of Islam—are written in Arabic. The *Qur'an* is the holy book of Islam, and *Hadith*[4] refers to the reports attributed to the Prophet Muhammad (peace be upon him), describing his words and actions. The Arabic language has been profoundly influenced by the *Qur'an* (Gholitabar and Kamali, 2015), which is regarded as the highest linguistic achievement in Arabic. Besides, Arabic is recognized as the fourth most used language on the web[5], with 237 million internet users and the fastest-growing language over the last five years.

[1]https://www.un.org/en/our-work/official-languages
[2]https://www.statista.com/statistics/806106/total-population-arab-league/
[3]https://timesprayer.com/en/muslim-population/
[4]https://www.oxfordbibliographies.com
[5]https://www.internetworldstats.com/

1.3.2 Arabic Varieties

Diglossia is a linguistic phenomenon in which two distinct varieties of the same language coexist within a community (Ferguson, 1972). Typically, one variety is considered "high" (formal and prestigious), while the other is regarded as "low" (informal and used in everyday conversation). Arabic is frequently cited by scholars as a leading example of diglossia due to its use of these distinct high and low varieties. Specifically, Arabic includes three main classes: Classical Arabic, Modern Standard Arabic (MSA), and Arabic dialects (Habash, 2010; Versteegh, 2014). In the context of diglossia, Classical Arabic and MSA represent the high varieties, whereas Arabic dialects constitute the low variety.

- **Classical Arabic:** Classical Arabic is a prestigious language within the Muslim world. It is the form of Arabic used in the Qur'an, *Hadith*, and classical literature, including poetry from before and shortly after the advent of Islam. Classical Arabic is distinguished by its complex grammatical and syntactical rules and serves as the foundation for Modern Standard Arabic.

- **Modern Standard Arabic (MSA):** Modern Standard Arabic is the official language used across the Arab world in media, education, and formal writing. However, MSA is primarily a written language and is not commonly spoken in everyday conversations. In contrast, the various Arabic dialects are used in daily life as the native spoken languages.

- **Arabic Dialects:** Arabic dialects, also known as Colloquial Arabic, are distributed across the 22 countries in the Arab world. According to Nizar Y. Habash(Habash, 2010), these dialects can be categorized into several groups: Egyptian Arabic (EGY), Levantine Arabic (LEV), Gulf Arabic (GLF), North African Arabic or Maghrebi (Mag), Iraqi Arabic (IRQ), and Yemeni Arabic (Yem). Maltese Arabic is considered a separate language, not an Arabic dialect, and is written in the Latin script. Arabic dialects are derived from MSA and exhibit many similarities with it. Arabic dialects are generally used in daily life as spoken languages. However, with the rise of social media, Arabic dialects are increasingly used in written forms such as comments and blogs. The authors in (Kathrein et al., 2018) explored the lexical connections between MSA and various Arabic dialects across different regions using natural language processing techniques. Their findings indicated that Levantine dialects are highly similar to one another, with the Palestinian dialect, which is a subset of the Levantine dialects, being particularly close MSA. MSA and Arabic dialects are sometimes written using Latin letters, a form known as Arabizi, Arabish, or Franco-Arab (Darwish, 2013). In Arabizi, numbers are used to represent phonetic sounds and letters that do not exist in

English or French. For example, "3" represents the letter "ع" and "7" represents the letter "ح" .Arabizi is commonly used in informal writing, such as on social media and in Short Message Service (SMS), and it is often mixed with English or French.

The variation in the Arabic language, combined with the widespread use of Arabizi, significantly complicates text classification tasks. Each Arabic dialect can differ in vocabulary, grammar, and even pronunciation, leading to inconsistencies in the data. When Arabizi is introduced into the mix, the classification becomes even more challenging, as the model must handle both the Arabic script and the informal Latin-based representation.

1.3.3 Arabic Script

The Arabic script is used for writing Arabic as well as several other languages from both the Asian and African continents that are not related to Arabic, such as Persian, Kurdish, and Urdu. It consists of the following 28 letters: أ ب ت ث ج ح خ د ذ ر ز س ش ص ض ط ظ ع غ ف ق ك ل م ن ه و ي. Unlike English and other Latin-based languages, Arabic is written from right to left and does not have uppercase or lowercase letters. The Arabic script utilizes two kinds of symbols: letters and diacritics (Habash, 2010):

- **Letters:** Letters in Arabic script are composed of two parts. The first part is the letter form, which is essential for each letter. The second part is the letter mark, which can be divided into three types: dots, the short Kaf, and the Hamza. Letter marks are used to distinguish different letters because some letters share the same letter form. For example, these three letters are differentiated by the number of dots: ب (/b/),ت (/t/), and ث (/θ/). Furthermore, Arabic letters may take four different shapes depending on their position in a word: at the beginning, in the middle, at the end, and when standing alone. For example, the letter ع(/ʕ/) can appear in the following shapes: Initial "عـ", Medial "ـعـ", Final "ـع", and Isolated " ع". Arabic letters are generally connected in cursive within words, except for certain letters (ا ، د ، ذ ، ر ، ز and و) that do not connect to the following letter. Long vowels in Arabic are represented by the letters: Alif "ا", Waw "و", and Ya "ي". Their corresponding phonetics are/aː/, /uː/, and /iː/ respectively.

Table 1.1 shows the Arabic alphabet with each letter's forms at the beginning, middle, and end of words, along with their pronunciation and the closest Latin equivalent.

- **Diacritics:** Arabic diacritics are represented by the three short vowels: Fatha (/a/), Damma (/u/), and Kasra (/i/). Additionally, there is the Sukun, which indicates the absence of these short vowels; Tanwin, which is pronounced like a short vowel followed by an /n/ sound at the end of the word; and Shadda, which represents a doubled consonant. Diacritics are optional because proficient speakers can understand a given text without them. Almost all Arabic texts are written without diacritics, which can lead to lexical ambiguity and pose challenges for computational systems (Boudad et al., 2017). For example, the absence of diacritics in the word بر may refer to بَرٌّ (land), بِرٌّ (charity), or بُرٌّ (wheat).

TABLE 1.1: Forms of Arabic letters.

Arabic Letter	**Beginning Form**	**Middle Form**	**End Form**	**Pronunciation**	**Closest Latin Equivalent**
ا	ا	ـا	ـا	ʔalif (a, ā)	A
ب	بـ	ـبـ	ـب	bāʔ (b)	B
ت	تـ	ـتـ	ت	tāʔ (t)	T
ث	ثـ	ـثـ	ـث	thāʔ (th)	TH
ج	جـ	ـجـ	ـج	jīm (j)	J
ح	حـ	ـحـ	ـح	ḥāʔ (ḥ)	H (hard H)
خ	خـ	ـخـ	ـخ	khāʔ (kh)	KH
د	د	ـد	ـد	dāl (d)	D
ذ	ذ	ـذ	ـذ	dhāl (dh)	DH
ر	ر	ـر	ـر	rāʔ (r)	R
ز	ز	ـز	ـز	zāy (z)	Z
س	سـ	ـسـ	ـس	sīn (s)	S
ش	شـ	ـشـ	ـش	shīn (sh)	SH
ص	صـ	ـصـ	ـص	ṣād (ṣ)	S (emphatic)
ض	ضـ	ـضـ	ـض	ḍād (ḍ)	D (emphatic)
ط	ط	ـط	ـط	ṭāʔ (ṭ)	T (emphatic)
ظ	ظ	ـظ	ـظ	thāʔ (ẓ)	DH (emphatic)
ع	عـ	ـعـ	ـع	ʔayn (ʕ)	ʔ (guttural stop)
غ	غـ	ـغـ	ـغ	ghayn (gh)	GH
ف	فـ	ـفـ	ـف	fāʔ (f)	F
ق	قـ	ـقـ	ـق	qāf (q)	Q
ك، ک	کـ	ـکـ	ـك، ـک	kāf (k)	K
ل	لـ	ـلـ	ـل	lām (l)	L
م	مـ	ـمـ	ـم	mīm (m)	M
ن	نـ	ـنـ	ـن	nūn (n)	N
هـ	هـ	ـهـ	ـه	hāʔ (h)	H
و	و	ـو	ـو	wāw (w, ū)	W
ي	يـ	ـيـ	ـي	yāʔ (y, ī)	Y

1.3.4 Arabic Morphology

In linguistics, morphology is the study of a word's internal structure and how words are formed (Aronoff and Fudeman, 2016) . Arabic morphology (علم الصرف العربي) is one of the main branches of Arabic grammar and is known for its richness and complexity, which makes tokenization and stemming processes more challenging compared to many other languages. Arabic morphology encompasses three main aspects: derivation, inflection, and agglutination (Habash, 2010; Boudad et al., 2017).

- **Derivation:** Derivation is the process of creating new words that are lexically related to a root. In English, new words are typically derived by adding a prefix or a suffix to the root. For example, from the English adjective happy, the adjective unhappy and the noun happiness are derived by adding a prefix and a suffix, respectively. Unlike English, Arabic morphology is non-concatenative (Farghaly et al., 2009). In Arabic, new words are not formed by simply adding prefixes or suffixes but by altering the word pattern. All Arabic words are based on a sequence of two, three, four, or five consonants known as the root. The arrangement of the root consonants with specific vowels and other consonants according to a particular pattern produces a variety of words, each generally conveying different concepts from the root. For example, from the Arabic root ك - ت - ب (K-T-B), which means 'write', several words can be derived:
    - The pattern 1a:2i3 produces the word كاتب (ka:tib), meaning 'writer', where the numbers indicate the position of the root consonants and the rest of the symbols indicate vowels and additional non-root consonants.
    - The pattern 1i2a:3 gives the word كتاب (kita:b), meaning 'book'.
    - The pattern Ma12a3 results in the word مكتب (maktab), meaning 'desk'.
- **Inflection:** Inflection is the process that involves a change in the form of a word to express different inflectional categories such as tense, number, gender, and others. Inflectional morphology differs from derivational morphology in that inflection produces different forms of the same word, whereas derivation produces entirely new words.

Compared to English, Arabic has a richer inflectional system. In fact, there are eleven inflectional categories in the Arabic language (Ryding, 2014):

1. **Tense/Aspect:** Tense (present, past, future); Aspect (imperfect, perfect).
2. **Person:** First, second, third.
3. **Voice:** Active, passive.
4. **Mood:** Indicative, subjunctive, jussive, imperative.
5. **Gender:** Masculine, feminine
6. **Number:** Singular, dual, plural.
7. **Case:** Nominative, genitive, accusative.
8. **Definiteness:** Definite, indefinite.
9. **Comparison:** Positive, comparative, superlative.
10. **Deixis:** Near, far, farther.
11. **Humanness:** Human/non-human distinction.

Table 1.2 presents the inflection of the verb فَعَلَ (to do) depending on person, number, gender, and tense.

TABLE 1.2: Inflection of the verb 'فعل' (to do).

	Personal pronouns (Ar.)	**Personal pronouns (En.)**	**Past tense**	**Transliteration**	**Present tense**	**Transliteration**
Single	أَنَا	I (male/female)	فَعَلْتُ	faealtu	أَفْعَلُ	afealu
	أَنْتَ	You (male)	فَعَلْتَ	faealta	تَفْعَلُ	tafealu
	أَنْتِ	You (female)	فَعَلْتِ	faealti	تَفْعَلِينَ	tafeali:n
	هُوَ	He	فَعَلَ	faeala	يَفْعَلُ	yafealu
	هِيَ	She	فَعَلَتْ	faealat	تَفْعَلُ	tafealu
Dual	نَحْنُ	We (male/female)	فَعَلْنَا	faealna:	نَفْعَلُ	nafealu
	انْتُمَا	You (male/female)	فَعَلْتُمَا	faealtuma:	تَفْعَلَانِ	tafela:n
	هُمَا	They(male)	فَعَلَا	faeala:	يَفْعَلَانِ	yafeala:n
	هُمَا	They(female)	فَعَلَتَا	faealata:	تَفْعَلَانِ	tafeala:n
Plural	نَحْنُ	We (male/female)	فَعَلْنَا	faealna:	نَفْعَلُ	nafealu
	انْتُم	You (male)	فَعَلْتُمْ	faealtum	تَفْعَلُونَ	tafealu:n
	انْتُنَّ	You (female)	فَعَلْتُنَّ	faealtunna	تَفْعَلْنَ	tafealnna
	هُم	They(male)	فَعَلُوا	faealu:	يَفْعَلُون	yafealu:n
	هُنَّ	They(female)	فَعَلْنَ	faealnna	يَفَعَلْنَ	yafealnna

- **Agglutination:** Arabic is an agglutinative language that relies primarily on clitics. Agglutination or cliticization have the same meaning, referring to the process of combining morphemes into a single word. Clitics have the form of affixes in that they are morphemes attached to a word, but they differ from affixes in their function. Unlike affixes, clitics depend phonologically on another word or phrase. For example, the auxiliary verbs *is* and *will* are reduced to the clitics *'s* and *'ll* in the phrases *I'm* and *I'll*, respectively. Table 1.3 represents Arabic clitics along with their corresponding classes, including conjunctions, particle proclitics, determiners, and pronominal enclitics. These clitics are attached to the stem in a strict order: CONJ + PART + DET + STEM + PRON. For instance, the Arabic word "وللمكتبات", which means "and for the libraries," can be split into four parts (و+ ل+ ال+ مكتبات): the conjunction "و" (and), the particle proclitic "لِ" (for), the determiner "ال" (the), and the stem "مكتبات" (libraries).

  The structure of Arabic words can be very complex; a single word in Arabic can function as an entire sentence in English. For example, the Arabic word "فَأَسْقَيْنَاكُمُوهُ" (13 letters) is equivalent to the fourteen-word English sentence "Then, we have made a provision in order to enable you to drink it" (51 letters). Another example is the word "أَنُلْزِمُكُمُوهَا" (10 letters), which corresponds to the seven-word English sentence "Shall we compel you to accept it?" (26 letters).

TABLE 1.3: Arabic clitics.

Cltics	Class	Function	English
أَ "a"	Conjunction	interrogative همزة الاستفهام	yes/no question
وَ "wa"	Conjunction	coordination واوالعطف	and
		connection واو الربط	and
		circumstantial واو الحال	while
	Particle proclitics	oath واو القسم	by
		Accompaniment واو المعية	with
فَ "fa"	Conjunction	coordination فاء العطف	and, so
		connection فاء الربط	and, so
		response conditional فاء الجزاء	so,then
		subordinating conjunction فاء السببية	so, that
بِ "bi"	Particle proclitics	Preposition حرف جر	by, with, in
كَ "ka"	Particle proclitics	Preposition حرف جر	such as, like
لِ "li"	Particle proclitics	Preposition حرف جر	to, for
لَ "la”		Emphasisلام التوكيد	will certainly
		response conditional لام الجزاء	so, then
سَ "sa"	Particle proclitics	future particle سين المستقبل	will
ال "al"	Determiner	definite article ال التعريف	the
ه "h"	Pronominal Enclitics	attached pronouns الضمائر المتصلة	His, its, him, it
ها "ha:"			Her, its, it, him
هم "hum"			Their, Them (Male, Plural more than 2)
هما "huma:"			Their, Them (Double)
هن "hunna "			Their, Them (Female, Plural more than 2)
ك "k"			Your, you (single)
كم "kum"			Your, you(Male, Pluralmore than2)
كما "kuma:"			Your, you (Double)
كن "kunna"			Your, you (Female, Plural more than 2)
نا "na:"			Our, us
ي "y "			My, me

1.3.5 Arabic Syntax

In linguistics, syntax refers to the rules that govern how words are arranged together to form phrases and sentences (Carnie, 2013). Arabic syntax (النَّحْو) is the second main branch of Arabic grammar, alongside morphology. Morphologically rich languages have a complex relationship between morphology and syntax. A prime example of this is the Arabic language, where Arabic cliticization morphology interacts with various syntactic structures (Habash, 2010). There are two types of Arabic sentences: verbal sentences and nominal sentences.A verbal sentence, "الجملة الفعلية," begins with the verb, followed by the subject. The defining characteristic of a verbal sentence is that the verb precedes the subject. The standard structure of verbal sentences is Verb-Subject-Object (فعل - فاعل - مفعول به). For example, "قرأ الطالب الكتاب" means "The student read the book." In this sentence, the verb is "قرأ" (read), the subject is "الطالب" (the student), and the object is "الكتاب" (the book). A nominal sentence, "الجملة الاسمية," begins with a noun and consists of a subject "مبتدأ" and a predicate "خبر." The subject is usually a definite noun, pronoun, or demonstrative, while the predicate can be an indefinite noun, adjective, or verb. For instance, in the nominal sentence "الجو جميل," which means "The weather is beautiful," the definite noun "الجو" (the weather) represents the subject, and the indefinite adjective "جميل" (beautiful) represents the predicate. Arabic syntax can be flexible, leading to various possible interpretations of sentence structure and meaning, which complicates the classification process.

1.4 Text Vectorization Techniques

Text documents are a form of unstructured data that need to be converted into numerical representations to be directly utilized by machine learning algorithms. This conversion process, known as text vectorization, is a fundamental challenge in text classification. Text vectorization techniques enable machine learning models to understand and analyze textual data by transforming words, phrases, or entire documents into vectors (numerical arrays).

1.4.1 One-Hot Encoding

One-Hot Encoding is a traditional technique used to encode qualitative data, such as textual words, into a numerical format suitable for machine learning models. In this approach, each distinct word in a text corpus is represented as a binary vector with a length equal to the number of unique words in the vocabulary. Within each vector, all positions are set to zero except for one position, which is marked with a "1" to indicate the presence of that specific word (Oguike, 2021). This method allows models to distinguish between words by their unique vector representations. Although one-hot encoding is appreciated for its simplicity, its main drawback is the creation of high-dimensional, sparse vectors, particularly with extensive vocabularies, which can be computationally costly. Additionally, since one-hot encoding does not capture semantic relationships between words, machine learning models that utilize these vectors as input may exhibit poor performance.

Consider a simple Arabic corpus consisting of two sentences:

1. "الجو جميل هذا اليوم" (The weather is beautiful today)
2. "الربيع فصل جميل" (Spring is a beautiful season)

This corpus contains six unique words indexed from 0 to 5:

Vocabulary={ "الجو" (weather): 0, "جميل" (beautiful): 1,"هذا" (this): 2, "اليوم" (today): 3, "الربيع" (spring): 4, "فصل" (season) : 5 }

Each of these words is represented by a binary vector with a length of 6:

"الجو"	=	[1, 0, 0, 0, 0, 0]
"جميل"	=	[0, 1, 0, 0, 0, 0]
"هذا"	=	[0, 0, 1, 0, 0, 0]
"اليوم"	=	[0, 0, 0, 1, 0, 0]
"الربيع"	=	[0, 0, 0, 0, 1, 0]
"فصل"	=	[0, 0, 0, 0, 0, 1]

Consequently, the sentence "الجو جميل هذا اليوم" is represented by the sequence of these vectors:

"الجو"	[[1, 0, 0, 0, 0, 0],
"جميل"	[0, 1, 0, 0, 0, 0],
"هذا"	[0, 0, 1, 0, 0, 0],
"اليوم"	[0, 0, 0, 1, 0, 0]]

Similarly, the sentence "الربيع فصل جميل" is represented by:

"الربيع"	[[0, 0, 0, 0, 1, 0],
"فصل"	[0, 0, 0, 0, 0, 1],
"جميل"	[0, 1, 0, 0, 0, 0]]

1.4.2 Bag-of-Words (BoW)

BoW model is a straightforward method for encoding text numerically by focusing on the frequency of words within a document, using a predefined vocabulary. Instead of treating the document as a sequence of words, it is considered a collection or "bag" of words (Jurafsky and Martin, 2023). The document is represented by counting how often each word from the vocabulary appears. For instance, if the vocabulary consists of 1,000,000 words, the document will be represented by a 1,000,000-dimensional vector, where each dimension corresponds to the frequency of a specific word. Typically, a word that appears frequently in a document is likely to convey a key idea about that document. This assumption is reasonable; for example, if the most common words in a document include "president," "voters," and "election," it is likely that the document is related to political topics. However, some words might be common across many documents and not be as specific to any single one. The term "bag" of words emphasizes that the model disregards the order and structure of words, focusing solely on the presence and frequency of vocabulary words, rather than their positions or sequence in the document.

The BoW model faces limitations including the need for careful vocabulary management to address sparsity and high dimensionality, and it ignores word order, which can hinder its ability to capture context and nuanced meanings. The N-grams method comes as an extension of the BoW model and has the ability to capture sequences of words, which helps in capturing some context and word order. However, using n-grams increases the complexity of the feature space.

1.4.3 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is a widely used method for vectorizing Arabic text and other languages. In (Spärck Jones, 1972), the author introduced the concept of inverse document frequency (IDF) to measure the specificity of terms in in a collection of documents.This concept later became a fundamental part of TF-IDF. The main idea behind TF-IDF is to assess the importance of a word in a document by considering how often it appears in the document (Term Frequency, TF) and how rare it is across the entire collection of documents (Inverse Document Frequency, IDF). Words that are common across many documents are assigned a lower weight, while those that appear in fewer documents receive a higher weight. This method highlights words that are significant to a specific document but less common across all documents. High dimensionality, sparsity, and lack of semantic understanding are the main drawbacks of TF-IDF.

The formula used to calculate the TF-IDF for a term t in a document d within a collection of documents is:

$$\text{tf-idf}(t, d) = \text{tf}(t, d) \times \text{idf}(t) \tag{1.1}$$

Where tf(t, d) represents the number of occurrences of term t in document d. The Inverse Document Frequency, idf(t, d), is represented by Equation 1.2, where N is the total number of documents in the corpus, and df(t) represents the number of documents that contain the term t. Note that adding a constant of 1 to the denominator is optional and prevents terms that appear in all training samples from receiving a zero value. The logarithm helps to limit the weight of terms with very low document frequencies.

$$\mathrm{idf}(t, d) = \log\left(\frac{N}{1 + \mathrm{df}(t)}\right) \tag{1.2}$$

1.4.4 Word Embedding

Word embeddings are a sophisticated method for representing text data as low-dimensional, dense vectors. Predominantly generated by advanced deep learning models, these embeddings are learned from extensive text corpora using unsupervised learning techniques, which do not rely on labeled data. By leveraging large-scale text data, word embeddings capture intricate semantic and syntactic relationships between words. This allows them to encode nuanced meanings, contextual information, and word associations within a continuous vector space. As a result, these vectors provide a compact and meaningful representation of words, enhancing the performance of algorithms in various NLP tasks such as text classification, sentiment analysis, and machine translation.

There are mainly two types of word embeddings: Static Word Embeddings and Contextual Word Embeddings.

1.4.4.1 Static Word Embeddings

Static word embeddings, also known as non-contextual embeddings, are pre-trained vectors representing words with fixed values that do not change based on the context in which the words are used. Prominent methods for static word embeddings include:

- **Word2Vec:** Word2Vec was introduced for the first time in (Mikolov et al., 2013) by Google. It utilizes neural networks to learn word representations from extensive text datasets. The main objective of Word2Vec is to capture the semantic relationships between words based on the contexts in which they appear. Word2Vec includes two primary models (see Figure 1.2): Continuous Bag of Words (CBOW) and Skip-gram. The CBOW model predicts a word based on its surrounding context. By using the surrounding words, CBOW focuses on predicting the target word and understanding word meanings by examining how often words co-occur. In contrast to CBOW, the Skip-gram model predicts the context words given a specific target word. This model is designed to capture more detailed semantic relationships by observing how words are used in various contexts. Word2Vec has notable drawbacks, including its inability to handle out-of-vocabulary (OOV) or rare words that were not seen during training. It also lacks sensitivity to morphological relationships, treating words with similar roots, such as "eat" and "eaten," as separate entities without leveraging their shared internal structure.

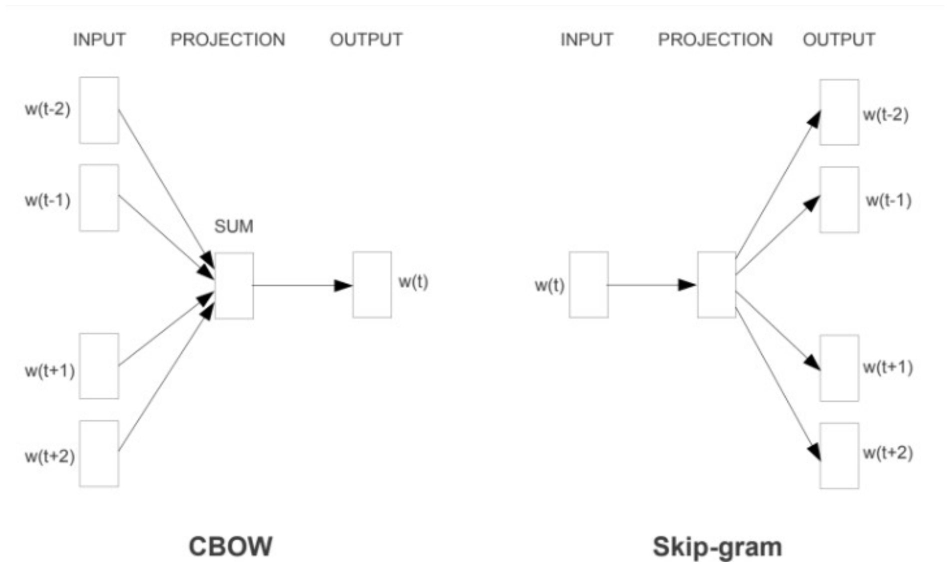


FIGURE 1.2: Word2Vec Models.

- **GloVe:** The Global Vectors for Word Representation, commonly known as GloVe, is a technique for learning word embeddings. it was developed by (Pennington, Socher, and Manning, 2014) at Stanford University, GloVe extends word2vec by combining both global and local text statistics. Unlike traditional methods such as Latent Semantic Analysis (LSA), which use matrix factorization for global statistics but lack contextual meaning, GloVe integrates these global statistics with local context-based learning. Rather than relying on a local window, GloVe constructs a word-co-occurrence matrix from the entire corpus. This approach not only reduces computational costs by employing a simpler least squares error function but also results in distinct and potentially improved word embeddings. Similar to Word2Vec, GloVe has limitations in dealing with out-of-vocabulary (OOV) words and morphological variations. It cannot produce embeddings for words that were not included in the training data and does not consider the internal structure of words with related roots.

- **fastText:** fastText (Mikolov et al., 2017) is an advanced word embedding technique developed by Facebook AI Research lab (FAIR) that addresses some limitations of earlier models such as Word2Vec and GloVe. By incorporating subword information, fastText enhances the handling of rare and out-of-vocabulary (OOV) words. It represents words as bags of character n-grams, allowing the model to capture morphological nuances and internal word structures. This approach enables fastText to generate embeddings for words not seen during training and to better manage variations in word forms, such as prefixes, suffixes, and inflections. As a result, fastText offers more robust and accurate word representations, improving performance on tasks involving complex word structures and languages with rich morphology. However, similar to Word2Vec and GloVe, fastText produces only a single vector representation per word, which does not account for its varying meanings across different contexts.

1.4.4.2 Contextual Word Embeddings:

The Transformer model (Vaswani et al., 2017) revolutionized NLP with self-attention mechanisms that capture complex text dependencies and weigh word importance, leading to more accurate and context-aware representations. Building on this, contextual word embeddings offer flexible vector representations for words that adapt according to their surrounding text, allowing the embeddings to adjust based on the specific context in which the words are used. Notable approaches for contextual word embeddings are:

- **BERT:** BERT, an acronym for Bidirectional Encoder Representations from Transformers (Devlin et al., 2019), is a language model based on deep learning techniques developed by Google. It uses transformers to dynamically calculate connections between input and output elements. Unlike traditional language models that process text in a single direction, BERT reads text bidirectionally, handling both left-to-right and right-to-left simultaneously, which enhances its ability to comprehend and process natural language. This approach significantly improved performance over previous models and established BERT as a key baseline in NLP. BERT surpasses Word2Vec, GloVe, and FastText by providing contextualized embeddings that adapt to word usage in different sentences, capturing nuanced meanings and polysemy that static models cannot. However, it does come with limitations, including high computational costs, a fixed input length constraint, and the need for extensive fine-tuning with labeled data to achieve optimal performance.

- **GPT:** GPT, which stands for Generative Pre-trained Transformer (Radford and Narasimhan, 2018), is a deep learning-based language model developed by OpenAI. It utilizes a Transformer decoder architecture to predict the next word in a sequence, processing text from left to right. Unlike bidirectional models like BERT, GPT focuses on generating coherent text by considering only the preceding context, which makes it particularly effective for text generation tasks. This approach has significantly advanced capabilities in creating fluent and contextually relevant content. However, GPT also has limitations, such as high computational requirements, constraints on input length, and generating text based only on past context without future considerations.

1.5 Dimensionality Reduction Techniques

Richard Bellman originally introduced the term "Curse of Dimensionality" while discussing dynamic programming (Bellman, 1957). This concept highlights the difficulties involved in analyzing and processing data in high-dimensional spaces. Text documents typically feature a wide array of unique words, each acting as a separate feature, resulting in high-dimensional data. This expansion in dimensions brings several challenges: it increases storage needs due to the larger feature set, intensifies computational complexity as more features require processing, and increases the risk of overfitting, where models might learn from noise or non-essential patterns rather than capturing useful trends. Employing dimensionality reduction methods, which involve decreasing the number of features, can help address these issues by improving both computational efficiency and model accuracy. Furthermore, dimensionality reduction simplifies complex high-dimensional data, making it more interpretable and enhancing visualization with clearer, lower-dimensional representations. Feature selection and feature extraction are the two primary types of dimensionality reduction methods used for text document.

1.5.1 Feature Extraction

Feature extraction, also known as feature transformation, is a feature reduction technique designed to create a new set of features by combining or transforming the original ones. This approach involves generating new, composite features that better capture the underlying patterns and relationships within the data compared to the original features. By reducing data to a lower-dimensional space, feature extraction techniques enhance visualization, thereby simplifying the analysis and interpretation of complex datasets. Various feature extraction methods grounded in mathematical and statistical principles, such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Non-Negative Matrix Factorization (NMF), are widely employed in text processing. These techniques aim to compress text documents while preserving their essential information.

1.5.1.1 Principal Component Analysis (PCA)

PCA, or Principal Component Analysis is a a widely-used unsupervised method for analyzing and visualizing high-dimensional data across various fields such as machine learning, computer vision, genomics, finance, speech processing, environmental science, and medical imaging. PCA reduces dimensionality by transforming high-dimensional data into a more compact, lower-dimensional representation while preserving as much of the original variance as possible (Jolliffe, 2011).

PCA is applied to text documents as follows: Consider a data matrix X where the rows correspond to documents and columns correspond to features (words or tokens). The first step is to standardize the data (by centering and normalizing) using Equation 1.3, where μ is the mean vector of the features and σ is the standard deviation vector for each feature. This step ensures that each feature has a mean of zero and a variance of one.

$$X_{\text{std}} = \frac{X - \mu}{\sigma} \tag{1.3}$$

The next step is to compute the Covariance Matrix as shown in Equation 1.4 where X_{std}^T is the transpose of the standardized data matrix. the Covariance Matrix displays the variance and the correlation among various features.

$$C = \frac{1}{m-1} X_{\text{std}}^T X_{\text{std}} \tag{1.4}$$

Now, obtaining eigenvalues and eigenvectors by performing eigen decomposition on the covariance matrix C, where eigenvectors($\mathbf{v}$) represent the principal components, and eigenvalues (λ) indicate the variance each principal component captures. The eigenvalue-eigenvector pairs are solutions to Equation 1.5 :

$$C\mathbf{v} = \lambda\mathbf{v} \tag{1.5}$$

then, sorting eigenvalues in descending order, selecting the corresponding eigenvectors, and choosing the top k eigenvectors (principal components) that capture the most variance, where k is based on the desired level of explained variance.

The final step is to project the original standardized data X_{std} onto the new feature space defined by the selected principal components, using the transformation Equation 1.6, where W is the matrix containing the top k eigenvectors as columns.

$$X_{pca} = X_{std}W \tag{1.6}$$

1.5.1.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA), which is sometimes referred to as Normal Discriminant Analysis or Discriminant Function Analysis is a supervised technique commonly employed for both classifying data and reducing data dimensionality. The general approach of LDA is similar to PCA but with an added focus. While PCA focuses on finding axes that maximize data variance, LDA also seeks axes that maximize the separation between multiple classes. LDA assesses two types of variance: within-class variance, which indicates how data points scatter within each class, and between-class variance, which gauges the distinction between classes (Sugiyama, 2007). Its goal is to reduce within-class variance while increasing between-class variance to improve class separation in the lower-dimensional space. To apply LDA to text documents, let consider a Matrix X of size $n \times d$, where n is the number of documents and d is the number of features (terms). Compute the mean vector for each class C_i as showed by Equation 1.7, where N_i is the number of documents in class C_i, and x represents a document vector in class C_i.

$$\mu_i = \frac{1}{N_i} \sum_{x \in C_i} x \tag{1.7}$$

For each class C_i, compute the Within-Class Scatter Matrix S_{W_i}:

$$S_{W_i} = \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T \tag{1.8}$$

Then, take the sum across all classes, where k is the number of classes:

$$S_W = \sum_{i=1}^{k} S_{W_i} \tag{1.9}$$

Compute the Between-Class Scatter Matrix:

$$S_B = \sum_{i=1}^{k} N_i(\mu_i - \mu)(\mu_i - \mu)^T \tag{1.10}$$

where μ is the overall mean vector of all documents:

$$\mu = \frac{1}{n} \sum_{x \in X} x \tag{1.11}$$

Solve the eigenvalue problem represented by Equation 1.12. Here, w denotes the eigenvectors and λ represents the eigenvalues:

$$S_W^{-1} S_B w = \lambda w \tag{1.12}$$

Choose the top k eigenvectors with the largest eigenvalues to form the projection matrix W. Finally, transform the original data matrix X into a lower-dimensional space as follows:

$$X_{\text{LDA}} = XW \tag{1.13}$$

1.5.2 Feature Selection

Feature selection refers to the technique of selecting a smaller, more relevant subset of features from the original set to effectively represent and classify textual data. This approach is more frequently examined and implemented in text processing compared to feature extraction. This approach simplifies the model and enhances its interpretability while maintaining or even improving the model's accuracy. While there are numerous feature selection methods available, some of the most commonly used in text processing include the Chi-Square (χ^2) Test, Mutual Information (MI), and Information Gain (IG).

1.5.2.1 Chi-Square (χ^2) Test

The chi-square (χ^2) test is a statistical method used to evaluate whether a specific feature, such as a word, is independent of the class labels (Yang and Pedersen, 1997). It assesses the difference between the expected frequency of the feature in each class and the observed frequency to determine if there is a significant relationship between the feature and the class labels.

The chi-square test can be computed using the formula:

$$\chi^2 = \sum_i \sum_j \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{1.14}$$

where O_{ij} represents the observed frequency of feature j in class i, and E_{ij} is the expected frequency. The expected frequency E_{ij} is computed as:

$$E_{ij} = \frac{\text{Total for row } i \times \text{Total for column } j}{\text{Grand Total}} \tag{1.15}$$

Although the Chi-Square test is straightforward and effective, it requires a large sample size to yield reliable results. Additionally, the test does not account for interactions between features or handle redundancy, which can limit its ability to capture complex patterns in text.

1.5.2.2 Mutual Information (MI)

Mutual Information or (MI) is a statistical measure that quantifies the amount of knowledge gained about one variable by knowing the value of another variable. It represents the reduction in uncertainty about one variable when information about another variable is available (Cover and Thomas, 2006). In the context of feature selection and text classification, MI helps determine the importance of a feature, such as a word or term, by measuring how much it reveals about the class labels, indicating its usefulness for distinguishing between different classes. terms with high MI values are viewed as more valuable and are typically chosen for inclusion in the model. A Zero MI signifies that the variables are independent, so knowing the value of one variable gives no information about the other.

Mutual Information between two variables X and Y is defined in Equation 1.16 The term $p(x, y)$ represents the joint probability distribution of X and Y, while $p(x)$ and $p(y)$ denote the marginal probability distributions of X and Y, respectively. This formula measures the relative entropy between the joint distribution and the product of the individual distributions $p(x)$ and $p(y)$.

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{1.16}$$

MI is advantageous because it is non-parametric, meaning it does not rely on any specific distribution for the variables, and it can capture non-linear relationships. However, it has limitations, including high computational complexity, especially with large datasets and numerous features, and the difficulty of accurately estimating joint and marginal probabilities, particularly with sparse data.

1.5.2.3 Information Gain (IG)

In (Quinlan, 1986), the author introduces the concept of Information Gain in the context of decision tree algorithms discussed later in Subsection 1.6.3. Information Gain is used to build decision trees by evaluating the information gain for each variable. The variable with the highest information gain is selected to split the dataset, which minimizes entropy and enhances the classification effectiveness.In text classification, Information Gain (IG) is used to assess how well a feature (such as a word) contributes to the classification of text documents. It assists in feature selection by evaluating the degree to which a feature distinguishes between different classes. Features with higher Information Gain are typically selected for inclusion in the classification model, as they contribute the most to reducing classification uncertainty.

Equation 1.17 calculates IG, which measures how much knowing a feature X reduces the uncertainty about a target variable Y. It represents the difference between the entropy of Y without X and the conditional entropy of Y given X. A high Information Gain indicates that X significantly reduces the uncertainty about Y.

$$G(Y \mid X) = H(Y) - H(Y \mid X) \tag{1.17}$$

IG is useful for identifying key features and is efficient in terms of computation, which makes it appropriate for handling extensive datasets. Nevertheless, it tends to favor frequently occurring words, which could result in less relevant features being selected. Additionally, IG does not consider the interactions between features, which might be important for contextual understanding, and it might include redundant features that convey similar information.

1.6 Classical Machine Learning-Based Approach

Machine Learning (ML), a field within artificial intelligence, is dedicated to allowing computers to learn from data and improve their ability to perform tasks without explicit programming. A specific type of ML, known as supervised learning, involves developing a prediction function called a "model" from labeled data. When these labels are discrete (qualitative variables), the process is referred to as classification. ML has revolutionized text classification by automating and improving the accuracy of textual analysis. Advanced algorithms like support vector machines and neural networks streamline the process, learning from large datasets to identify complex patterns. This enables effective handling of tasks such as sentiment analysis, spam detection, and topical classification, while reducing the time and cost associated with manual methods.

Classical machine learning uses traditional algorithms and statistical methods to create predictive models from data, relying on predefined features and straightforward model structures. These methods are effective and interpretable for tasks like text classification. Despite the rise of more complex techniques, classical machine learning continues to provide valuable solutions for various problems.

1.6.1 Logistic Regression (LR)

LR is a supervised classification technique used to predict the probability of a categorical outcome based on a set of independent variables. Unlike linear regression, which is applied to quantitative outcomes, LR is suited for scenarios where the dependent variable is qualitative. There are two main types of logistic regression: binary logistic regression and multinomial logistic regression. Binary logistic regression is employed when the outcome variable has two categories, whereas multinomial logistic regression is used when the outcome variable has more than two categories.The Logistic Function, also known as the sigmoid function, is central to LR and is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1.18}$$

where z is the linear combination of input features, typically represented by equation 1.19 in the case of multiple independent variables or by equation 1.20 for a single independent variable.

$$z = a_1x_1 + a_2x_2 + \cdots + a_nx_n + b \tag{1.19}$$

$$z = ax + b \tag{1.20}$$

1.6.2 *k*-Nearest Neighbors (*k*NN)

KNN algorithm is widely regarded as one of the most straightforward machine learning algorithms (Cover and Hart, 1967). Unlike other algorithms, *k*NN does not involve a training phase to learn a discriminative function from the data. Instead, it relies on storing the entire dataset. The fundamental principle of the *k*NN method is as follows: when classifying a data point with an unknown class, the algorithm compares it to all stored data using a distance metric. The class assigned to the new data point is determined by the majority class among its *k* nearest neighbors. The effectiveness of the *k*NN algorithm is influenced by the choice of distance metric and the parameter *k*, which represents the number of neighbors considered. Typically, the Euclidean distance measure is used to identify the *k* nearest neighbors of a data point. In n-dimensional Euclidean space, the Euclidean distance d between two data points represented by vectors $v = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ is given by:

$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{1.21}$$

1.6.3 Decision Trees (DT)

A decision tree is a machine learning technique employed for both classification and regression that splits data based on decision rules, forming a tree-like structure with nodes and branches. It starts with a root node and divides the dataset through internal nodes based on feature values, continuing until reaching leaf nodes that provide the final decision. Decision trees use splitting metrics such as information gain or Gini impurity for classification and variance reduction for regression. They are known for their simplicity and interpretability but can suffer from overfitting and instability. Additionally, decision trees serve as the foundation for more advanced methods like Random Forests. Figure 1.3 shows an example of a decision tree for rain forecasting, with outcomes indicating whether it will rain (Yes) or not rain (No)

1.6.4 Support Vector Machine (SVM)

SVMs (Vapnik, Golowich, and Smola, 1996) are one of the most powerful supervised machine learning techniques, widely used for classification problems, including text classification. However, it can also be employed to solve regression problems. The core concept of SVM is to find a hyperplane that optimally separates the data while maximizing the margin between the data points and the hyperplane.This margin is the distance between the hyperplane and the nearest data points from each class, known as support vectors (see Figure 1.4).

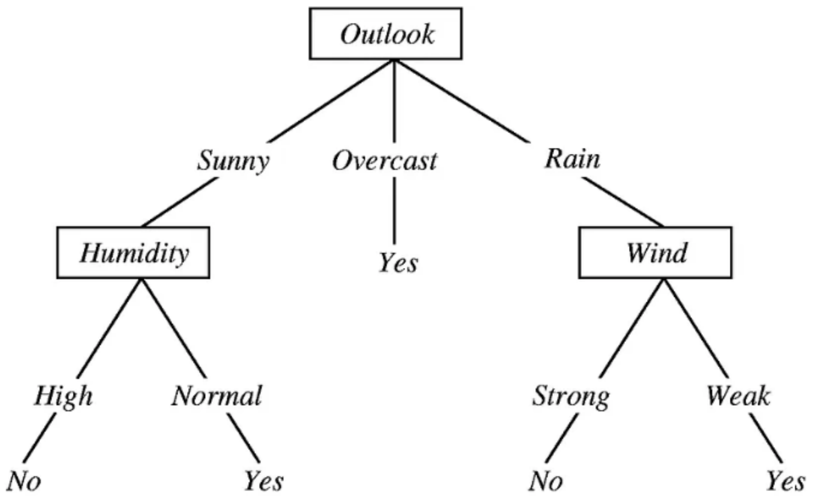


FIGURE 1.3: Decision tree example for rain forecasting.

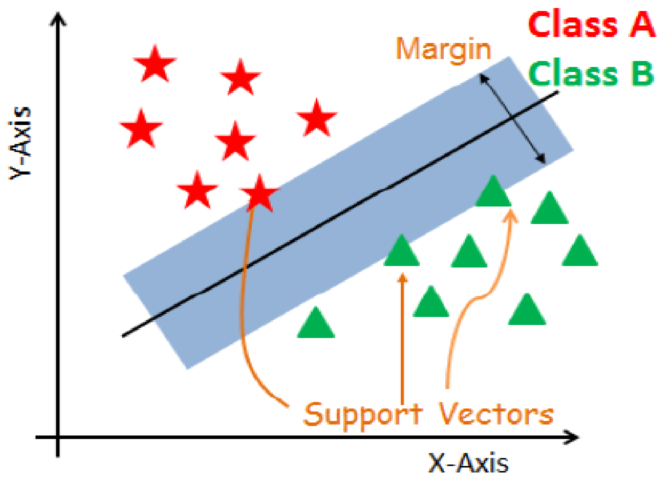


FIGURE 1.4: Example showing the support vectors and the margin.

By maximizing this margin,the generalization error is minimized for better classification performance. As Figure 1.5 shows, the right plot demonstrates that with an optimal hyperplane, a new sample (data point) remains well-classified even if it falls within the margin. In the left plot, it is observed that with a smaller margin, the sample is poorly classified.

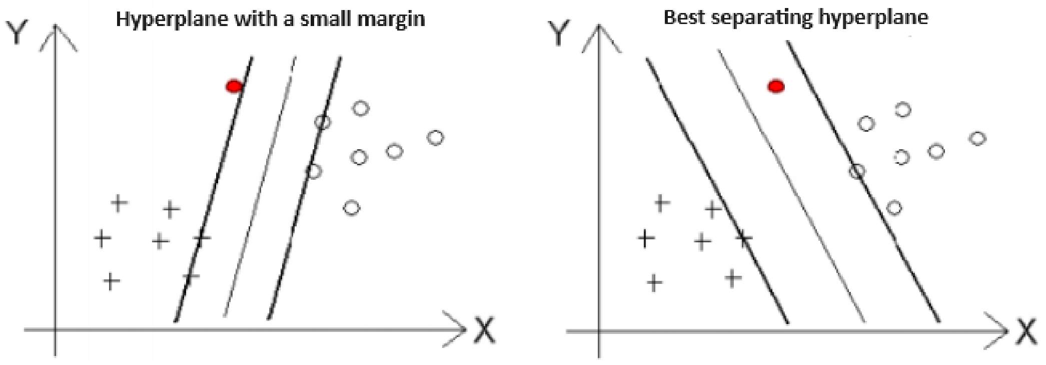


FIGURE 1.5: Example illustrating poor and good margins.

SVMs are highly effective in high-dimensional spaces, making them well-suited for tasks involving a large number of features, such as text classification. They perform well with linearly separable data and are also versatile enough to handle non-linearly separable data by employing kernel functions, such as the radial basis function (RBF) or polynomial kernels, which map the data into a higher-dimensional space where a linear separation becomes possible. Additionally, SVMs include a regularization parameter, denoted as C, which helps control overfitting by balancing the trade-off between maximizing the margin and minimizing classification errors.

1.7 Deep Learning-Based Approach

Deep learning is a subfield of machine learning that focuses on algorithms inspired by the structure and function of the brain, known as artificial neural networks. It involves using neural networks with multiple layers, referred to as deep neural networks, where the term "deep" signifies the presence of numerous layers within the network. In contrast to classical machine learning, deep learning models excel at automatic feature extraction from raw and large datasets, often requiring substantial computational resources. Deep learning models have set new benchmarks across various fields, particularly in numerous natural language processing (NLP) applications. In text and document classification, four primary types of deep learning models are commonly used, including Convolutional Neural Networks, Recurrent Neural Networks, Attention Mechanisms and Transformers.

1.7.1 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a type of deep learning architecture that have been effectively used for text classification. Although they were originally developed for image processing, specifically for handwritten digit recognition (LeCun et al., 1989). The term "convolutional neural network" refers to its use of convolution, which is a specialized mathematical linear operation. Figure 1.6 illustrates a convolutional neural network (CNN) architecture designed for classifying Arabic text. The process starts with encoding the Arabic text into an input matrix, which transforms the text into a numerical format that the network can process. Subsequently, various filters are applied to this matrix to detect different features within the text. This filtering process generates feature maps that represent the presence of these detected features. To make the model more efficient and robust, max pooling is employed to reduce the dimensionality of the feature maps, while retaining the most significant features. Finally, a softmax layer is used to convert the processed features into probabilities for different classes, allowing for accurate text classification.

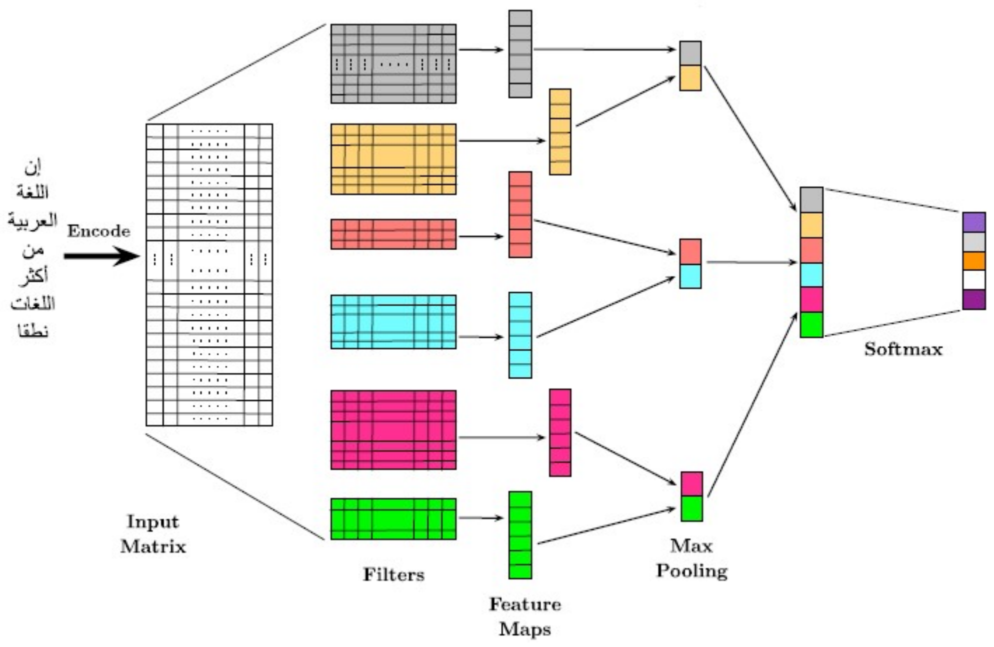


FIGURE 1.6: Example of a CNN architecture for Arabic text classification (Alhawarat and Aseeri, 2020).

1.7.2 Recurrent Neural Network (RNN)

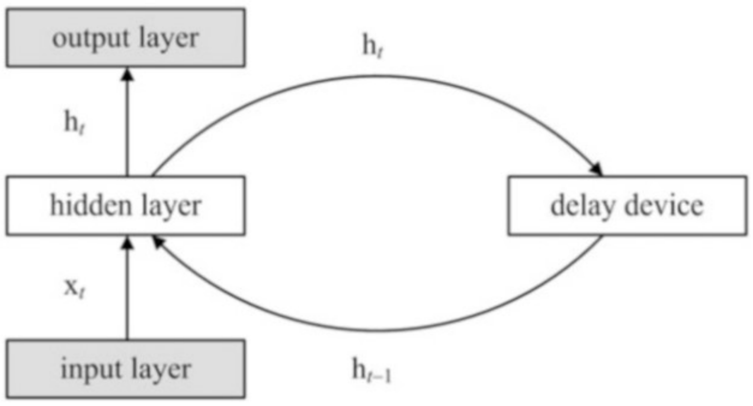


FIGURE 1.7: Basic architecture of RNN

Recurrent neural networks , also known as RNNs (Rumelhart, Hinton, and Williams, 1986) are a type of deep learning architecture designed to handle sequential or time series data, addressing the limitations of traditional feedforward networks that struggle with retaining historical information. The core component of an RNN is the "recurrent unit", which maintains a hidden state or memory. This hidden state is updated at each time step based on the current input and the previous hidden state, allowing the network to learn from past inputs and apply that knowledge to current processing. This capability makes RNNs particularly well-suited for tasks involving sequences, such as text. Figure 1.7 illustrates a basic RNN architecture. In this setup, the input layer receives data at each time step (x_t) and passes it to the hidden layer. The hidden layer processes this input along with the previous hidden state (h_{t-1}), which is maintained by a delay device. This mechanism allows the network to retain information across time steps. The current hidden state (h_t) is then used to generate the output. The relationship between the hidden states and the inputs is described by Equation 1.22 where (h_0) is initialized to zero and f can be either a nonlinear function or a feedforward network. This architecture enables the RNN to capture temporal dependencies in sequential data.

$$h_t = f(h_{t-1}, x_t) \tag{1.22}$$

Traditional RNNs struggled with the vanishing gradient problem, where gradients used to update the network's weights become very small during training. This makes it difficult for the network to learn and remember information from earlier in the sequence, limiting its ability to capture long-range dependencies. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks are advanced types of RNNs designed to address this problem and are often used for text classification tasks.

- **LSTM :** Long Short-Term Memory , often abbreviated as LSTM, is a type of RNN developed in 1997 (Hochreiter and Schmidhuber, 1997) designed to address the limitations of traditional RNNs, particularly the vanishing gradient problem, which affects their ability to learn long-term dependencies in sequential data. LSTM networks are built around the cell state, which acts as a memory to retain long-term information across sequences. This cell state is updated through three types of gates: the forget gate, which decides which information to discard; the input gate, which determines what new information to add; and the output gate, which controls what information from the cell state should be output as the hidden state. These gates work together to regulate the flow of information, enabling LSTMs to effectively maintain and utilize long-term dependencies. LSTMs act as essential components within RNN layers, managing data through "weights" to determine whether new information should be retained, forgotten, or given sufficient importance to impact the output. The LSTM architecture has demonstrated significant effectiveness in text classification, particularly when used in conjunction with other deep learning methods, such as attention mechanisms and pre-trained language models. However, LSTMs have limitations, including increased computational complexity, difficulties with very long sequences, and a tendency to overfit, particularly with limited data.

- **GPU:** Gated Recurrent Units (GRUs) (Chung et al., 2014) is an extension of LSTMs, introduced seventeen years after LSTMs were first developed. GRUs simplify the LSTM architecture by eliminating the explicit cell states. Unlike LSTMs, which use separate forget and output gates to manage the flow of information, GRUs use a single reset gate and an update gate to achieve a similar effect. This simplification makes GRUs generally faster and computationally less expensive. Despite these differences, the fundamental concept of GRUs closely resembles that of LSTMs, particularly in how they partially reset the hidden states. As Figure 1.8 depicts, a GRU differs from an LSTM in that it has only two gates and does not include internal memory (e.g. C_{t-1} in Figure 1.8). It also omits the second non-linearity, such as $\tanh(x)$ function, found in LSTMs.

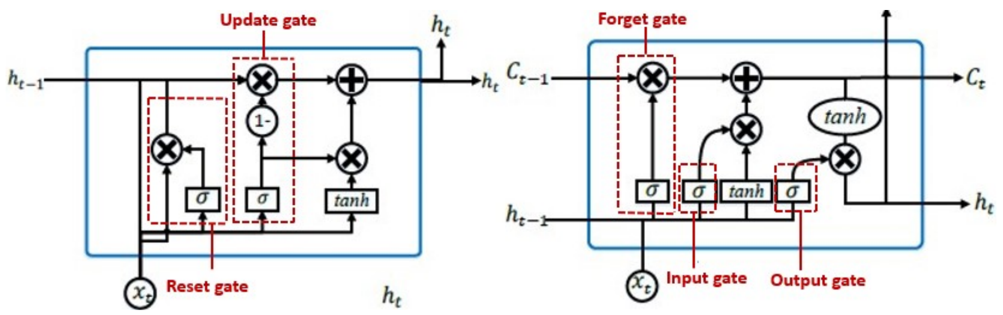


FIGURE 1.8: The GRU cell is on the left and the LSTM cell is on the right.

- **Bidirectional RNN:** Bidirectional Recurrent Neural Networks (BiRNNs) (Schuster and Paliwal, 1997) are commonly used for text classification tasks, including models such as Bidirectional Long Short-Term Memory (BiLSTM) and Bidirectional Gated Recurrent Unit (BiGRU). These models process sequential data by analyzing input sequences in both forward and backward directions, allowing them to leverage contextual information from both the past and the future. Traditional Recurrent Neural Networks (RNNs) process sequences in a single direction, typically forward, which limits their ability to capture context from the entire sequence. In contrast, a BiRNN employs two separate recurrent hidden layers: one that processes the sequence in the forward direction and another that processes it backward (see Figure 1.9). The outputs of these layers are combined and fed into a final prediction layer. The hidden layers in BiRNNs can be constructed using various recurrent cells, such as LSTM or GRU. The forward hidden layer updates its state based on the current input and the previous state, while the backward hidden layer updates its state based on the current input and the subsequent state. This bidirectional approach enhances the model's performance and provides additional regularization.

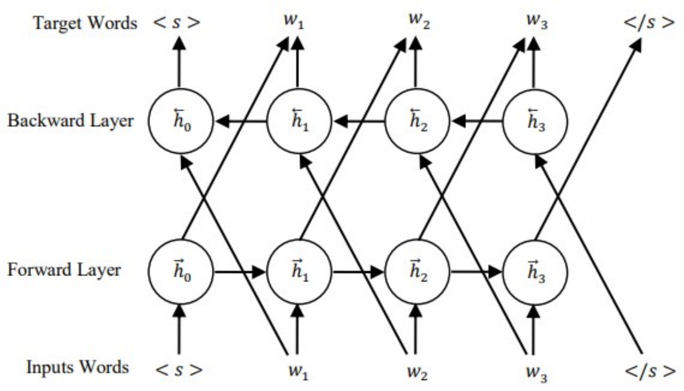


FIGURE 1.9: Bi-directional Recurrent Neural Network.

1.7.3 Attention Mechanism

The concept of attention in machine learning is inspired by biological attention, where humans selectively focus on relevant information rather than processing all available data. This principle, observed across various sensory perceptions, is applied in machine learning fields such as computer vision and natural language processing to enhance performance by concentrating on the most pertinent data. The attention mechanism was first introduced in (Bahdanau, Cho, and Bengio, 2014) and initially applied to machine translation. In text classification, attention mechanisms can be incorporated into RNNs to improve model performance by allowing the model to focus on significant parts of the input text when making predictions. Instead of treating all words equally, attention assigns different weights to words based on their relevance to the classification task, helping the model capture context and nuances more effectively and improving text classification accuracy.

1.7.4 Transformers

The authors of (Vaswani et al., 2017) presents a new deep learning architecture known as the Transformer. Transformers use self-attention mechanisms to process sequences in parallel, allowing them to capture long-range dependencies and relationships between words efficiently. The architecture consists of an encoder and a decoder (see Figure 1.10) , each with multiple layers that include multi-head attention and feed-forward networks, text is first converted into numerical tokens, which are then mapped to vectors using a word embedding table. At each layer, tokens are contextualized within a context window through a parallel multi-head attention mechanism, enhancing key tokens while diminishing less important ones.

Transformers offer the advantage of not using recurrent units, which reduces training time compared to earlier recurrent neural networks (RNNs) like LSTMs. This efficiency has led to their widespread adoption in training large language models (LLMs) such as BERT and GPT on extensive datasets.

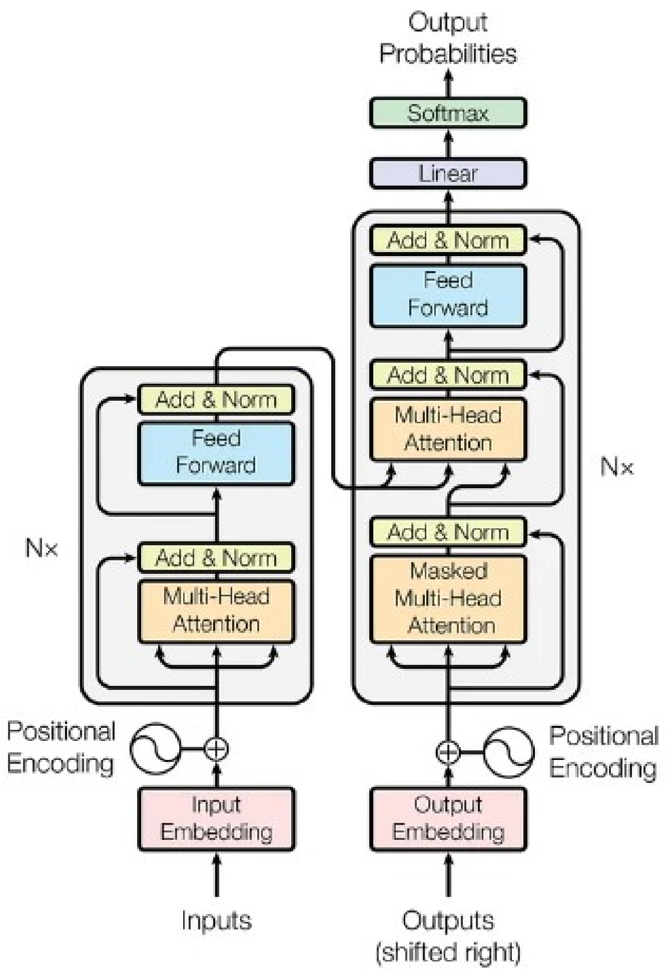


FIGURE 1.10: The Transformer architecture, with the encoder on the left and the decoder on the right (Vaswani et al., 2017).

1.8 Related Works

Although the majority of document classification research has been focused on the English language, Arabic language applications have been relatively overlooked. To address this gap, many studies have tackled Arabic document classification using a variety of datasets, data preprocessing methods, vectorization techniques, dimensionality reduction methods, classification approaches, and performance evaluation metrics.

1.8.1 Datasets

Data collection is a critical first step in building a text classifier for NLP applications. While English document text classification benefits from a range of freely available benchmark datasets, such as 20 Newsgroups and Reuters 90. In contrast, Arabic document classification datasets are more limited and relatively smaller compared to English datasets. Various Arabic datasets have been used in Arabic Text Classification (ATC) , with most collected from Arabic news websites. The sizes of these datasets range from 1,000 to 450,000 documents, classified into 3 to 28 categories. Most Arabic Text Classification (ATC) datasets focus on single-label classification, with only a few addressing multi-label problems (Alalyani and Marie-Sainte, 2018). Many datasets lack a standardized classification scheme, although the NADA dataset (Alalyani and Marie-Sainte, 2018) stands out for using the Dewey Decimal Classification. Some studies have used imbalanced datasets, which causes classifiers to focus on the majority classes due to their higher frequency during training, often resulting in reduced performance when classifying the minority classes.; however, techniques like the Synthetic Minority Over-Sampling Technique (SMOTE) have been applied in other works (Al-Azani and El-Alfy, 2017; Alalyani and Marie-Sainte, 2018) to balance dataset classes and improve accuracy. The study by (Ababneh, 2022) evaluated seven Arabic text classification datasets using five models. It found that the SVM model, when trained on the Khaleej and Arabiya datasets (both subsets of the SANAD dataset) (Einea, Elnagar, and Debsi, 2019), achieved the highest accuracy and shortest training time. This finding provides valuable guidance for researchers in selecting effective datasets for ATC. Table 1.4 provides a summary of the datasets used in Arabic document classification research.

TABLE 1.4: MSA document datasets.

Ref.	**Dataset Name**	**Size**	**Dataset Type**	**Num. of Classes**	**Class Names**	**Classification Type**
(Wahbeh and Al-Kabi, 2012)	N.A.	1,000	Balanced	4	sports, politics, economics, and *Hadith*	Single-label
(Al khurayji and Sameh, 2017)	N.A.	1,897	Balanced	3	Economic, culture and sport	Single-label
(Saad and Ashour, 2010)	BBC Arabic corpora	4,763	Imbalanced	7	Middle East News, World News, Business & Economy, Sports, International Press, Science & Technology, and Art & Culture	Single-label
(Saad and Ashour, 2010)	CNN Arabic corpora	5,070	Imbalanced	6	Business, Entertainments, Middle East News, Science & Technology, Sports, and World	Single-label
(Al Qadi et al., 2019)	N.A.	89,189	Balanced	4	Business, Sports, Technology, and Middle East	Single-label
(Abbas and Smaïli, 2005)	Khaleej-2004	5,690	Imbalanced	4	Economy, International News, Local News, and Sport	Single-label
(Saad and Ashour, 2010)	OSAC	22,429	Imbalanced	10	Economics, History, Entertainments, Education & Family, Religious & Fatwas, Sports, Heath, Astronomy, Low, Stories, Cooking Recipes	Single-label
(Einea, Elnagar, and Debsi, 2019)	SANAD	200,000	Imbalanced	7	Finance, Sports, Culture, Tech, Politics, Medical, and Religion	Single-label
(Elnagar, Al-Debsi, and Einea, 2020)	NADiA	451,230	Imbalanced	30	Leaders, Sports, Arabian Sports, Football Clubs,Arts, Cancer, Technology, Religion, Islamic, Fatawa, Worship, etc.	Multi-label
(Biniz et al., 2018)	N.A.	111,728	Imbalanced	5	sport, politic, culture, economy , and diverse.	Single-label
(Al-Tahrawi and Al-Khatib, 2015)	Al-Jazeera News	1,500	Balanced	5	Art, Economic, Politics, Science, and Sport	Single-label
(Abuaiadah, El-Sana, and Abusalah, 2014)	DDA	2,700	Balanced	9	Art, Literature, Religion, Politics, Law, Economy, Sport, Health, and Technology.	Single-label
(Alalyani and Marie-Sainte, 2018)	NADA	13,066	Balanced	10	Social science – economy, Social science – politics, Social science – law, General Religions – Islam, Applied science – computer science, etc.	Single-label
(Essma and Guessoum, 2015)	TALAA	57,827	Imbalanced	8	Culture, Economics, Politics, Religion, Society, Sports,World, Other.	Single-label

1.8.2 Text Preprocessing

The goal of data preprocessing in text documents is to eliminate noisy and unnecessary features, thereby reducing resource demands and enhancing classification accuracy. There is no standardized preprocessing method in the field of ATC. Tokenization is a fundamental step applied across all text classification tasks. Most researchers implement common preprocessing steps, such as removing digits, punctuation marks, non-Arabic text, and stop words (see 1.5). Some also remove diacritics (Elnagar, Al-Debsi, and Einea, 2020; Mahmoud and Zrigui, 2019; Alwehaibi and Roy, 2018; Lulu and Elnagar, 2018; Baali and Ghneim, 2019; Al-Tahrawi and Al-Khatib, 2015) and elongation (Elnagar, Al-Debsi, and Einea, 2020; Lulu and Elnagar, 2018). In certain cases, punctuation marks are treated as individual words (Abdullah, Hadzikadic, and Shaikh, 2018). Normalization of specific Arabic letters is often avoided in many studies (Al khurayji and Sameh, 2017; Alshammari, 2018; Qadi et al., 2019; Sundus, Al-Haj, and Hammo, 2019; Elnagar, Al-Debsi, and Einea, 2020; Mahmoud and Zrigui, 2019; Lulu and Elnagar, 2018; Abdullah, Hadzikadic, and Shaikh, 2018; Biniz et al., 2018) due to the potential risk of altering the contextual meaning of certain words. However, some researchers have chosen to apply normalization despite this concern (Wahbeh and Al-Kabi, 2012; Alwehaibi and Roy, 2018; El-Alami and Alaoui, 2016; Baali and Ghneim, 2019).

Extracting stems (stemming) or lemmas (lemmatization) is a key preprocessing step that reduces the feature set size. Lemmatization, while more accurate, is harder to implement for morphologically complex languages like Arabic and is slower, as it requires predefined dictionaries to identify the lemma. In contrast, stemming involves simply removing affixes from words, making it faster but less accurate. Despite this, the reduced accuracy is often acceptable for document classification and other NLP tasks (Elbarougy, Behery, and El Khatib, 2020). Consequently, stemming is commonly used in ATC research (Duwairi, 2014; Al khurayji and Sameh, 2017; Alshammari, 2018; Al-Tahrawi and Al-Khatib, 2015; Sundus, Al-Haj, and Hammo, 2019; El-Alami and Alaoui, 2016; Biniz et al., 2018; Baali and Ghneim, 2019; Ayed, Labidi, and Maraoui, 2017). Despite the trend in recent NLP literature to overlook preprocessing, a recent study on English text classification (Siino, Tinnirello, and La Cascia, 2024) explores its effect on both modern pre-trained Transformers and traditional models. The research applies three leading preprocessing techniques to four datasets from different text classification tasks and tests nine models, including Transformers. Results show that while both Transformers and traditional models benefit from preprocessing, simple models can outperform the top-performing Transformer with the right preprocessing strategy.

1.8.3 Text Vectorization

Text documents, being unstructured data, must be transformed into numerical representations to be effectively used by machine learning algorithms. In the last decade, many works in ATC have utilized high-dimensional vectorization techniques, including TF-IDF, which is the most commonly used method (Wahbeh and Al-Kabi, 2012; Al khurayji and Sameh, 2017; Al Qadi et al., 2019; Sundus, Al-Haj, and Hammo, 2019; Biniz et al., 2018), as well as the Boolean model (Alshammari, 2018), Bag of Words (BoW) (El-Alami and Alaoui, 2016), and N-grams (Al-thubaity, Alhoshan, and Hazzaa, 2015; Ayed, Labidi, and Maraoui, 2017). These models struggle with large and complex datasets due to their inability to capture semantic relationships, and they produce inefficient, sparse, and high-dimensional representations. Recent research focuses on low-dimensional, dense textual data vectorization using word embeddings, including word2Vec (Elnagar, Al-Debsi, and Einea, 2020; Mahmoud and Zrigui, 2019; Baali and Ghneim, 2019), AraVec (Mohammed and Kora, 2019; Sagheer and Sukkar, 2018; Alwehaibi and Roy, 2018) which is introduced in (Soliman, Eissa, and El-Beltagy, 2017), an open-source project that provides free, pretrained word embeddings for Arabic NLP research, implemented using the Word2vec model. AraVec includes various iterations based on skip-gram and continuous bag of words (CBoW) methodologies, GloVe (Mahmoud and Zrigui, 2019), and Arabic FastText(Alwehaibi and Roy, 2018; Alghamdi and Assiri, 2019). Other studies have employed contextual word embeddings, such as the Arabic BERT (AraBERT) model (El-Alami, Alaoui, and En-Nahnahi, 2021), for Arabic text multi-class classification, utilizing it both as a transfer learning model and as a feature extractor with different classifiers. The findings reveal that fine-tuned AraBERT achieves state-of-the-art performance.

1.8.4 Text Dimensionality Reduction

For effective and efficient Arabic text classification, dimensionality reduction is vital because of the language's complex morphology and large feature space. A range of methods has been suggested to boost classification performance while addressing the challenges associated with high-dimensional data. Many Feature selection techniques are used in the ATC. (Abu-Arqoub, Issa, and Hadi, 2019) explored the impact of Chi-square (χ^2), Information Gain (IG), and Correlation-based Feature Selection (CFS) on classifiers such as SVM, NB, kNN, and DT using Arabic datasets from the Saudi Press Agency (SPA). The results showed that feature selection generally improves classification accuracy by eliminating irrelevant features. Another paper (Elhassan and Ali, 2019) examined the impact of Information IG and Chi-square on Arabic text classifiers using NB and Sequential Minimal Optimization

(SMO) models. It found that feature selection generally improves classifier performance, with IG outperforming Chi-square for the NB classifier and showing similar results with the SMO classifier. (Bahassine et al., 2020) presented an improved Chi-square feature selection method (ImpCHI) for ATC. It compared ImpCHI with traditional metrics (mutual information, information gain, and standard Chi-square) and evaluates its performance using an SVM classifier on a dataset of 5,070 Arabic documents. The results showed that ImpCHI combined with SVM significantly enhances classification accuracy, achieving a best F-measure of 90.50% with 900 features.(Alshaer, Otair, Abualigah, et al., 2021) confirmed the effectiveness of ImpCHI over Chi-square across six classifiers, including Random Forest, Decision Tree, Naïve Bayes, Naïve Bayes Multinomial, Bayes Net, and Artificial Neural Networks—using a dataset of 9,055 Arabic documents and various evaluation measures. (Chantar et al., 2019) presented a feature selection method for ATC using the Hybrid Binary Gray Wolf Optimizer (HBGWO). Combining Binary Gray Wolf Optimizer (BGWO), Particle Swarm Optimization (PSO), and Salp Chain Algorithm (SCA), HBGWO improved classification accuracy to 88.08% on a dataset of *Hadith* books, outperforming other methods and BGWO-PSO alone. The author (Atef Mosa, 2022) combines a Knowledge-Graph with Ant Colony Optimization (ACO) to predict semantic categories in Arabic *Hadith* texts. The KG links features to categories based on their co-occurrence, and ACO selects the most relevant features. Using over 30,000 *Hadith* from six books and nearly 120 categories, the approach improved classification accuracy by 3% when integrated with machine learning classifiers. (Hadni and Hjiaj, 2023) addressed the challenge of ATC by proposing a chaotic sine cosine-based Firefly Algorithm (FA), which combines chaos theory with FA for feature selection. Using the Kalimat dataset and SVM classifiers, the model improved classification performance by 2% compared to the standard FA.

Other research works have utilized feature extraction techniques in Arabic document classification to transform the original features into a more compact set in order to enhance classification effectiveness. (Al-Anzi and AbuZeina, 2017) used Singular Value Decomposition (SVD) to extract Latent Semantic Indexing (LSI) features from TF-IDF. A comparison of classification methods, including Naïve Bayes, k-Nearest Neighbors, Neural Networks, Random Forest, Support Vector Machines, and Classification Trees, was conducted using a corpus of 4,000 documents across ten topics. The results show that classification methods leveraging LSI features significantly outperformed those based on TF-IDF. (Al-Taani and Al-Sayadi, 2020) introduced an approach for ATC using singular value decomposition (SVD) and fuzzy c-means. The effectiveness of the method is evaluated on Al Jazeera and CNN Arabic news datasets. The results are compared with four supervised classification techniques including support vector machine, naive Bayes, decision tree, and polynomial networks—previously applied to the same datasets. The findings demonstrate that the proposed approach outperforms recent methods in Arabic text classification. Similarly, Singular Value Decomposition (SVD) is proposed in (Harrag and Al-Qawasmah, 2010) to reduce the dimensionality of the input data and enhance the effectiveness and efficiency of the neural network. Experiments on an Arabic corpus of *Hadith* show that the NN model with SVD performs better than the basic NN model, achieving higher precision, recall, and F-measure. (El-Alami and Alaoui, 2016) presented a deep learning method for Arabic Text Categorization (ATC) using deep stacked autoencoders to improve text representation and reduce dimensionality. Experiments on a CNN Arabic news dataset with Light and Khoja stemmers showed that the method, combined with Decision Tree, Naïve Bayes, and Support Vector Machine techniques, achieved strong performance in categorizing Arabic text.

Upon reviewing previous research on incorporating dimensionality reduction steps after text vectorization in the Arabic text classification (ATC) pipeline, and despite the limited number of such studies compared to those on English language text, we observed that most dimensionality reduction methods applied to ATC primarily use selection methods and overlook transformation methods. Additionally, the majority of these studies focus solely on effectiveness while neglecting efficiency, which is a key advantage of dimensionality reduction techniques. The performance of these studies indicates potential for improvement. Moreover, many rely on only one small dataset or, at most, two datasets, which risks overfitting and limits the evaluation of how well the approach generalizes to other data types or scenarios.

1.8.5 Classical Machine Learning and Deep Learning Models

In the last decade, research in Arabic text classification has predominantly focused on applying machine learning algorithms such as Support Vector Machines (SVM), Naive Bayes (NB), k-Nearest Neighbors (kNN), Decision Trees, and Logistic Regression. (Wahbeh and Al-Kabi, 2012) compared the performance of three classifiers: Support Vector Machine (SVM), Naïve Bayes (NB), and decision tree based on C4.5 algorithms for for Arabic text classification. The study found that Naïve Bayes achieves the highest accuracy, while SVM is the quickest to build the model, followed by Naïve Bayes and C4.5. (Al khurayji and Sameh, 2017) propose a Kernel Naive Bayes (KNB) classifier to address the non-linearity issue in textual data and compare it with baseline classifiers such as Naïve Bayes (NB), Hidden Markov Model (HMM), Support Vector Machines (SVM), k-Nearest Neighbors (kNN), and J48. Experimental results show that the KNB classifier outperforms the other baseline classifiers on an Arabic topic mining corpus. (Alshammari, 2018) emphasized the importance of preprocessing techniques for preparing datasets for machine learning algorithms.

The study also introduced DMNBtext, a new variant of Naïve Bayes for text classification. When compared with traditional Naïve Bayes and the C4.5-based decision tree, DMNBtext outperforms the other classifiers on two Arabic datasets, achieving 99% accuracy on the BBC dataset and over 93% accuracy on the CNN dataset. The authors (Al Qadi et al., 2019) developed a new dataset consisting of 89,189 Arabic news articles, categorized into four classes: Business, Sports, Technology, and Middle East. This dataset was evaluated using ten supervised machine learning classifiers, including LR, Nearest Centroid, DT, SVM, KNN, XGB, RF, Multinomial NB, Ada-Boost, and MLP. Among these, SVM achieved the highest performance with an F1-score of 97.9%.

The breakthrough of deep learning in the fields of computer vision and natural language processing (NLP) for English motivated researchers in ATC to follow this trend. Inspired by the significant improvements in accuracy and performance achieved by deep learning models like convolutional neural networks (CNNs) and transformers, ATC researchers began exploring similar techniques to overcome the unique challenges posed by Arabic language processing. (Sundus, Al-Haj, and Hammo, 2019) presented a feed-forward deep learning neural network for Arabic text classification. The model's first layer uses TF-IDF vectors based on the most frequent words in the document collection, which serve as input to the second layer. To minimize classification errors, Adam's optimizer was applied. Experiments were conducted on two multi-class Arabic datasets, evaluating the model using metrics like precision, recall, F-measure, support, accuracy, and model building time. Results showed that the deep learning approach outperformed logistic regression in Arabic text classification tasks. (Elnagar, Al-Debsi, and Einea, 2020) introduced two new datasets for Arabic text categorization: SANAD for single-label tasks and NADiA for multi-label tasks, both of which were made freely available to researchers. The study compared several deep learning models without requiring pre-processing, including BIGRU, BILSTM, CGRU, CLSTM, CNN, GRU, HANGRU, HANLSTM, and LSTM, while also exploring the use of word2vec embeddings. Attention-GRU achieved the highest performance on both the SANAD and NADiA datasets. CNN models was design in (Sagheer and Sukkar, 2018) to classify Arabic sentences into three categories using the Essex Arabic Summaries Corpus (EASC). The CNN models utilized a word embedding layer, either pre-trained or learned during training, and incorporated dropout and L2 weight regularization to combat overfitting. They achieved high accuracy in classifying Arabic sentences. (Biniz et al., 2018) introduced a novel approach to Arabic text classification. It starts by using an Arabic stemming algorithm to extract, select, and reduce relevant features. Term Frequency-Inverse Document Frequency (TF-IDF) is then applied for feature weighting. For the classification step, the study employs Convolutional Neural Networks (CNNs), a powerful algorithm commonly used in image processing and pattern recognition but less explored in text mining. By combining these techniques and fine-tuning CNN hyperparameters, the method achieves outstanding results on various benchmarks.

1.8.6 Evaluation

Experimental evaluation is a crucial step in the testing phase, used to measure the generalization ability of a trained classifier, i.e., how accurately the model can predict outcomes on new, unseen text. It is also employed during the training phase to optimize the classifier. Several performance metrics can assess the quality of text classifiers, including accuracy, precision, recall, and the F1-score. In ATC research, accuracy is the most commonly used metric (Wahbeh and Al-Kabi, 2012; Al khurayji and Sameh, 2017; Alshammari, 2018; Al Qadi et al., 2019; Sundus, Al-Haj, and Hammo, 2019; Elnagar, Al-Debsi, and Einea, 2020; Sagheer and Sukkar, 2018; Mahmoud and Zrigui, 2019; Alwehaibi and Roy, 2018; Lulu and Elnagar, 2018; El-Alami and Alaoui, 2016; Biniz et al., 2018; Baali and Ghneim, 2019; Mohammed and Kora, 2019), while precision, recall (Duwairi, 2014) and F1-score (Bahassine et al., 2020) are used less frequently. A major limitation of accuracy is that it fails to capture information about false negatives and false positives. On the other hand, precision overlooks true negatives and false negatives, whereas recall does not account for true negatives and false positives (Kowsari et al., 2019). Choosing the appropriate evaluation metric is essential for optimizing the classifier and improving text classification performance. Some studies do not specify which evaluation metric was used to assess their models (Abdullah, Hadzikadic, and Shaikh, 2018; Al-Tahrawi and Al-Khatib, 2015). To the best of our knowledge, no study has specifically examined evaluation metrics for ATC.

Table 1.5 provides an overview of these studies by detailing key aspects for each research work. It includes information on the dataset used, the Data Preprocessing techniques applied, the Vectorization methods and Dimensionality Reduction techniques utilized, the model employed, how the dataset was divided into training and testing sets, and the performance achieved.(Note: N.A. indicates "Not Available").

TABLE 1.5: A summary of a related works in Arabic document Classification.

Ref.	Dataset	Preprocessing	Vectorization	Dimensionality Reduction	Model	Train \Test (%)	Best Model	Evaluation
(Wahbeh and Al-Kabi, 2012)	News articles (1,000 document)	-Removing digits, punctuation marks, non-Arabic text, and. stop words -Normalization -Tokenization	TF-IDF	No	-SVM -NB -C4.5	- 60\40 - 10-folds CV	NB	Accuracy (85.25%)
(Al khurayji and Sameh, 2017)	News articles (1,897 document)	-Removing stop words -Tokenization . -Stemming (light stemmer)	TF-IDF	No	-Kernel Naïve Bayes (KNB) -NB -HMM -SVM -KNN -J48	70\30	KNB	Accuracy (91.20%)
(Alshammari, 2018)	-BBC -CNN	-Document conversion -Removing stop words Tokenization -Stemming: -light stemmer -Khoja stemmer -no stemming	-boolean -idf,tf,tfidf -tfidf-norm minFreq3 -tfidf-norm-minFreq5 -wc -wc-minFreq3 -wc-minFreq5 -wc-norm -wc-norm-minFreq3 -wc-norm-minFreq5	No	-NB -Discriminative Multinominal Naive Bayes for text (DMNBtext) -C4.5	10-folds CV	DMNBtext	Accuracy 99% on BBC 93% on CNN
(Al Qadi et al., 2019)	News articles (89,189 document)	-Removing stop words, Latin characters, numbers, and punctuation	TF-IDF	No	-LR -Nearest Centroid -DT -SVM -KNN -XGB -RF -Multinomial NB -Ada-Boost -MLP.	80\20	SVM	Accuracy (94.4%)

Ref.	Dataset	Preprocessing	Vectorization	Dimensionality Reduction	Model	Train \Test (%)	Best Model	Evaluation
(Elnagar, Al-Debsi, and Einea, 2020)	-SANAD -NADiA	-Removing non-Arabic content, diacritics, elongation, punctuation marks, extra spaces	word2vec	No	-CNN -LSTM, BILSTM, CLSTM, HANLSTM -GRU, BIGRU, CGRU, HANGRU	80\10 and 10 for validation	HANGRU	Accuracy (96.94 % on SANAD 88.68 % on NADiA
(El-Alami and Alaoui, 2016)	CNN	-Removing stop word, punctuation marks,numbers,non-Arabic content -Normalization -Tokenization -Stemming:Light stemmerand Khoja stemmer	BoW	deep stacked autoencoder	-DT -NB -SVM	N.A.	SVM	Accuracy (75.10 %
(Biniz et al., 2018)	News articles(111,728 document)	-Removing stop words, foreign characters, punctuation, and numbers -Stemming	TF-IDF	Eliminate terms with very low scores	-CNN -LR -SVM	70\30	CNN	Accuracy (92.94 %)
(Al-Taani and Al-Sayadi, 2020)	-CNN -Al-Jazeera News	-Removing stop words, non-Arabic characters, numbers, special characters -Tokenization stop-words removal, elimination of non-Arabic	Entropy-Based Document-Term Matrix	SVD	-FCM	N.A.	FCM	F-measure: 61.11% on CNN 76% on Al-Jazeera News

Ref.	Dataset	Preprocessing	Vectorization	Dimensionality Reduction	Model	Train \Test (%)	Best Model	Evaluation
(Sundus, Al-Haj, and Hammo, 2019)	-Khaleej-2004 -News articles (1,445 document)	-Removing stop words -Tokenization -Stemming	TF-IDF	No	-feed forward supervised DL -LR	80\20	feed forward supervised DL	Accuracy 93.80 % on Khaleej-2004 94.12 % on News articles
(Al-Tahrawi and Al-Khatib, 2015)	Al-Jazeera News	-Tokenization -Removing stop words, non-Arabic content, numbers, diacritics, pecial characters and punctuations -Stemming (Khoja stemmer)	N.A.	Chi Square	-PN -SVM -NB -KNN -DT(J48)	80\20	SVM	Weighted Average (93.70 %)
(Sagheer and Sukkar, 2018)	EASC	Tokenization	AraVec	No	- CNN	75\25 50\50	CNN	Accuracy (75%)
(Bahassine et al., 2020)	CNN	-Tokenization -Removing stop words, non-Arabic characters, numbers, special characters and punctuations -Stemming -Normalization	TF-IDF	-ImpCHI -Chi-square -MI -GI	-SVM -DT	80\20	ImpCHI + SVM	F-measure (90.50%)
(Al-Anzi and AbuZeina, 2017)	News articles (4000 document)	-Tokenization -Removing stop words, non-Arabic characters, numbers, special characters -Normalization	TF-IDF	LSI	-SVM -DT -kNN -SVM -RF LR	90\10	SVM	Accuracy (84.75%)

Ref.	Dataset	Preprocessing	Vectorization	Dimensionality Reduction	Model	Train \Test (%)	Best Model	Evaluation
(Abu-Arqoub, Issa, and Hadi, 2019)	News articles (1,526 document)	N.A.	N.A.	-CFS -IG -Chi-Square	-SVM -kNN -DT -NB	N.A.	SVM	Accuracy (73 %)
(Elhassan and Ali, 2019)	News articles (1,000 document)	-Removing stop words, non-Arabic characters, numbers, special characters and punctuations -Stemming	N.A.	-IG -Chi-Square	-SMO -NB	10-fold CV	Chi-Square + NB	N.A.

1.9 Conclusion

In conclusion, this chapter has laid a solid foundation for the research by presenting key concepts and challenges in Arabic document classification, with a specific focus on dimensionality reduction. The examination of the unique characteristics of the Arabic language and their impact on classification has provided valuable insights into the complexities of this task. Furthermore, the review of both classical and deep learning models, alongside dimensionality reduction techniques, highlights the significance of simplifying and optimizing data to enhance classification performance. This chapter sets the stage for the following chapters, where the contributions of dimensionality reduction to Arabic document classification will be further elaborated and analyzed in detail.

Chapter 2

Arabic Text Classification Using Principal Component Analysis With Different Supervised Classifiers

2.1 Introduction

Various traditional machine learning approaches have been employed for text classification tasks (Kowsari et al., 2019). To prepare textual data for classifiers, it is often converted into high-dimensional numerical feature vectors. One of the most established methods for this numerical representation is term frequency-inverse document frequency (TF-IDF), which quantifies the importance of each word in the context of a given document relative to a corpus. However, a high dimensionality of features can lead to increased computational complexity and memory requirements, and may also result in overfitting, thereby reducing the effectiveness of many classifiers (Skillicorn, 2012). To address these challenges, techniques such as feature selection and feature extraction are frequently employed to enhance model performance and efficiency. In feature selection, a subset of the most significant features is selected from the original set to represent the data effectively. An extensive review of various feature selection methods can be found in (Abdallah and La Iglesia, 2015). In contrast, feature extraction, also known as feature transformation, involves creating a new set of features by combining or transforming the original ones (Ghojogh et al., 2019). One of the most recognized feature extraction methods is Principal Component Analysis (PCA), which has been widely utilized across different fields for dimensionality reduction and data analysis (Jolliffe, 2002). PCA demonstrated improvements in both accuracy and processing time for clustering Arabic and English documents (Abozied, 2019) and also enhanced the classification of Persian texts (Zahedi and Sorkhi, 2013).

In this chapter, we present a revised version of our conference paper presented in (Louail, Kara-Mohamed Hamdi-Cherif, and Hamdi-Cherif, 2021), where we utilize Principal Component Analysis for feature extraction in Arabic text classification and evaluate its influence on the effectiveness and efficiency of several prominent classifiers, such as support vector machines, random forests, decision trees, k-nearest neighbors, and logistic regression. By incorporating PCA, we aim to address the challenges associated with high-dimensional data in Arabic text processing, which enhances both computational efficiency and model performance. To the extent of our knowledge, this is the first instance of PCA being applied to Arabic text classification. The remainder of the chapter is structured as follows: Section 2.2 details the materials and methods employed, along with the implementation of the proposed approach. Section 2.3 presents the results and provides a discussion of the findings. Finally, Section 2.4 offers a conclusion.

2.2 Materials and methods

2.2.1 Proposed system architecture

As Figure 2.1 shows, our system architecture begins by preprocessing the dataset, which includes cleaning the text and converting it into numerical features using the TF-IDF representation. The preprocessed dataset is then split into training and testing sets. Principal Component Analysis (PCA) is fitted on the training set to reduce dimensionality. The fitted PCA is subsequently used to transform both the training and testing sets. The transformed training set is used to train a machine learning classifier, resulting in a model. This trained model is then applied to the transformed testing set to generate predictions and evaluate performance.

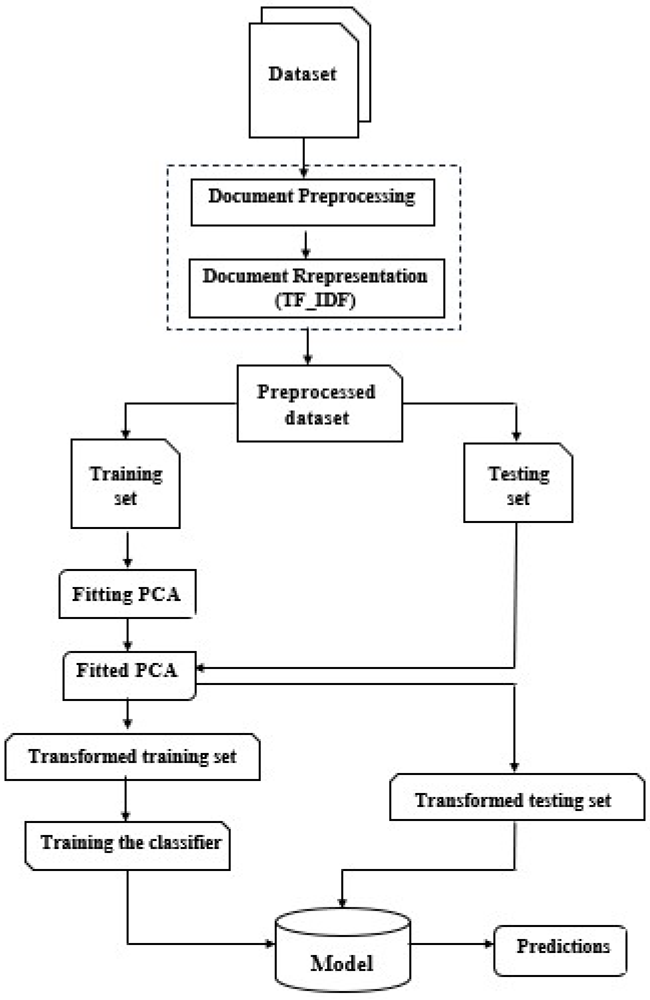


FIGURE 2.1: Proposed system architecture.

2.2.2 Datasets

To evaluate the performance of Principal Component Analysis (PCA) in conjunction with various classifiers, we employed two publicly available Arabic text datasets: a subset of the SANAD dataset (also known as the AlArabiya dataset) and the NADA dataset.

1. **AlArabiya:** The AlArabiya corpus is a central dataset within the SANAD collection (Einea, Elnagar, and Debsi, 2019) and is highly applicable for ATC as well as a range of other NLP applications. For our experiments, we used 1,500 documents from the AlArabiya dataset. These documents were evenly distributed across five categories: finance, sports, medicine, politics, and technology.

2. **NADA:** the NADA dataset (Alalyani and Marie-Sainte, 2018) was created by improving two previously existing datasets, OSAC (Saad and Ashour, 2010) and Diab Dataset

(DAA). The enhancements involved several preprocessing steps, reorganizing the data using the Dewey Decimal Classification system, and applying a synthetic minority over-sampling technique to address class imbalances. The NADA dataset comprises 7,310 documents classified into ten categories, such as computer science, economics, health sciences, Islam, law, literature, politics, sports, art, and astronomy. For our study, we utilized a subset of 1,500 raw documents from this dataset.

- **Dataset Splitting** The dataset was split into two subsets: a training set and a testing set. The training set was used to train the machine learning models, while the testing set was used to evaluate their performance. A common split ratio of 80/20 was applied, where 80% of the data was allocated to the training set and 20% to the testing set. This split ensured that the model was trained on a substantial amount of data while retaining enough data for a robust evaluation.

Table 2.1 provides a summary of the datasets and their specifics as utilized in our study.

TABLE 2.1: Description of the Datasets

Dataset	**No. Classes**	**Classes**	**No. Docs.**	**No. Docs./ Class**
AlArabiya	5	Finance, Medicine, Politics, Sports and Technology.	1,500	300
NADA	10	Arabic Literature, Islam, Economy, Politics, Law, Computers, Health, Astronomy, Art and Sport.	1,500	150

2.2.3 Document text preprocessing

Text preprocessing is a crucial step in preparing documents for effective analysis by cleaning them of unnecessary data before encoding. This phase involves a series of traditional preprocessing operations that are essential for improving the quality of the data and the performance of subsequent models. After testing various approaches, we determined that the preprocessing pipeline outlined below greatly boosted the effectiveness of training for the machine learning classifiers and increased the accuracy of the outcomes, while simultaneously reducing the size of the vocabulary:

- **Data cleaning:** This step involved removing digits, special symbols, and non-Arabic characters to ensure that the text data consisted solely of relevant Arabic text, thus avoiding confusion and inaccuracies in subsequent processing. Additionally, stop-words—common words that do not contribute significant meaning, such as conjunctions حروف العطف and prepositions—were eliminated. Arabic conjunctions, which link words, phrases, or clauses, and prepositions حروف الجر, which indicate relationships between nouns or pronouns and other words, were among those removed. Examples of Arabic conjunctions include و (and), أو (or), and لكن (but), while common prepositions include في (in), على (on), and من (from). We utilized a comprehensive Arabic stop-words (Zerrouki, 2010) list containing 13,629 entries, which enabled us to effectively filter out a wide range of uninformative words. Examples of removed stop-words include إذما (if),and إذن (then).

- **tokenization:** The Camel_tools tokenizer [1] was utilized for splitting the texts into individual words and separating certain punctuation marks or symbolic characters. The Camel_tools tokenizer is part of the CAMeL Tools suite, developed by the CAMeL Lab at New York University Abu Dhabi. This suite provides a comprehensive range of Arabic natural language processing (NLP) tools designed to support various linguistic tasks.

- **stemming:**We used the ISRI Arabic stemmer from the Natural Language Toolkit (NLTK) library [2] to reduce words to their root forms, standardizing the text and enhancing data consistency. This preprocessing step is crucial in morphologically rich

[1]https://camel-tools.readthedocs.io/en/latest/api/tokenizers/word.html
[2]https://www.nltk.org/api/nltk.stem.html

languages like Arabic, as it simplifies words by removing affixes, helping to handle various word forms effectively.

2.2.4 Document text representation

After preprocessing, the documents are converted into numerical vectors using TF-IDF. This technique quantifies the importance of words in text classification. Term Frequency (TF) measures how often a term appears in a document, while Inverse Document Frequency (IDF) evaluates the term's uniqueness across the dataset. By multiplying TF and IDF, the TF-IDF score highlights terms that are significant within a document but less common across the entire corpus. This transformation results in each document being represented as a vector of TF-IDF scores, facilitating further analysis and enabling text classification.

2.2.5 Dimentionality reduction using PCA

Principal Component Analysis is used to reduce the dimensionality of the TF-IDF vectors, transforming the data into a more manageable form while preserving its essential information. The process begins by fitting the PCA model to the training set. This involves calculating the principal components from the features of the training data. Specifically, PCA analyzes the variance-covariance matrix of the training data to identify the principal components—directions in which the data exhibits the greatest variance. By projecting the data onto these principal components, PCA effectively reduces the dimensionality while preserving as much of the original variance as possible. Once the principal components are determined, they are used to transform both the training and testing datasets into a lower-dimensional space. This transformation ensures that both datasets are aligned in the same reduced-dimensional space.

2.2.6 Classifiers used and hyperparameter tuning

To evaluate the effect of reducing the features using PCA method, we have trained five well-known classifiers, namely LR, kNN, DT, RF and SVM. All these algorithms with their implementations, are available via the scikit-learn[3] online free platform. Hyperparameter tuning involves adjusting pre-set parameters of a machine learning model to enhance its performance. Unlike model parameters, which are learned during training, hyperparameters are configured before training begins and influence the learning process. We employed grid search, which exhaustively explores a specified set of hyperparameter values, combined with

[3]https://scikit-learn.org/stable/index.html

a 5-fold cross-validation strategy to evaluate the performance of various hyperparameter settings for kNN, DT, RF and SVM classifiers, as shown in Table 2.2.

- **kNN**: The hyperparameter n_neighbors controls the number of neighbors considered in classification, with values ranging from 17 to 35 to be tested (odd values are preferred to avoid ties). We adopted the common rule of thumb $k = \sqrt{N}$, where N is the number of training samples. This approach helps balance the risks of overfitting (when k is too small) and underfitting (when k is too large). For smaller datasets, dividing by 2 is often recommended for further refinement. With our training set of 1,200 samples, this suggests a k value of approximately 17, consistent with the lower end of the range. The weights parameter defines the influence of neighbors on the classification: 'uniform' means all neighbors contribute equally, while 'distance' means closer neighbors have a stronger influence, with their contribution weighted by distance.

- **DT:** DT Hyperparameters include the criterion parameter, which determines how the quality of a split is measured. It can be set to 'gini', which uses Gini impurity to assess the likelihood of incorrect labels, or 'entropy', which uses information gain to evaluate the randomness or uncertainty in the dataset.

- **RF:** RF classifier is optimized using two hyperparameters: n_estimators,which indicates the number of decision trees in the forest, tested at values of 50 and 100. Generally, a greater number of trees tends to improve model performance by averaging errors and minimizing overfitting; however, it also leads to increased computational costs, and criterion, which measures split quality using either Gini impurity or entropy.

- **SVM:** we shoose a linear SVM classifier (kernel = 'linear'), which means no transformation of the data, and the model will seek a linear decision boundary. The parameter C ranges from [0.001, 0.01, 0.1, 1, 10, 100] and controls the regularization strength. Smaller C values lead to a wider margin and more regularization, potentially increasing misclassifications, while larger C values reduce regularization and fit the training data more closely.

- **LR:** Default hyperparameters are used for LR, with an 'l2' penalty (Ridge regularization) is applied to prevent overfitting by adding the square of the coefficients to the loss function, encouraging smaller coefficients. The regularization strength is controlled by the parameter C, with C = 1 balancing bias and variance; smaller values of C increase regularization, while larger values reduce it.

TABLE 2.2: Classifiers' hyperparameters.

Classifier	Hyperparameter

LR	Penalty=l2
	C=1

KNN	n_neighbors=[17,19,21,23,25,27,29,31,33,35]
	weights=['uniform','distance']

DT	criterion=['gini', 'entropy']

RF	n_estimators=[50,100]
	criterion=['gini','entropy']

SVM	kernal='linear'
	C=[0.001,0.01,0.1,1,10,100]

2.2.7 Implementation

Our system was implemented in a Python 3 environment using Google Colab, which offered a flexible and scalable platform for development and experimentation. We used the Scikit-learn library for a range of tasks, including text preprocessing, representation, feature extraction through Principal Component Analysis, and for training and evaluating the performance of five different classifiers.

2.3 Results and Discussions

1. **Cumulative explained variance (CEV):**

    - **Use of AlArabiya dataset:** Figures 2.2 presents the cumulative explained variance percentage as a function of the number of components for the AlArabiya dataset. The cumulative explained variance increases steadily and gradually, indicating that the dataset has a more evenly distributed variance across a large number of components. Notably, it requires around 1000 components to account for nearly 100% of the variance, suggesting that the dataset possesses a complex structure with numerous dimensions contributing to its overall variance.

    - **Use of NADA dataset:** Figure 2.3 illustrates the cumulative explained variance for the NADA dataset, where the variance rises sharply at first and then flattens more quickly, indicating that a significant portion of the variance is captured by the first few hundred components. Fewer components, around 800, are needed to explain nearly 100% of the variance in this case, suggesting that the NADA dataset has a few dominant principal components and a more compact structure in terms of variance distribution.

- **Comparative Analysis on both datasets:** The comparative analysis reveals key differences between the AlArabiya and NADA datasets. In terms of complexity, the AlArabiya dataset exhibits greater complexity with a more evenly distributed variance, requiring more components to capture the same amount of variance compared to the NADA dataset. On the other hand, the NADA dataset demonstrates greater efficiency in dimensionality reduction, as fewer components are needed to capture the majority of the variance, suggesting more pronounced principal components or a more structured form.

  These differences have implications for tasks like feature selection, dimensionality reduction, or data visualization. The NADA dataset may allow for a more significant reduction in dimensionality without significant information loss, while the AlArabiya dataset might require more careful handling to avoid losing important variance during dimensionality reduction.

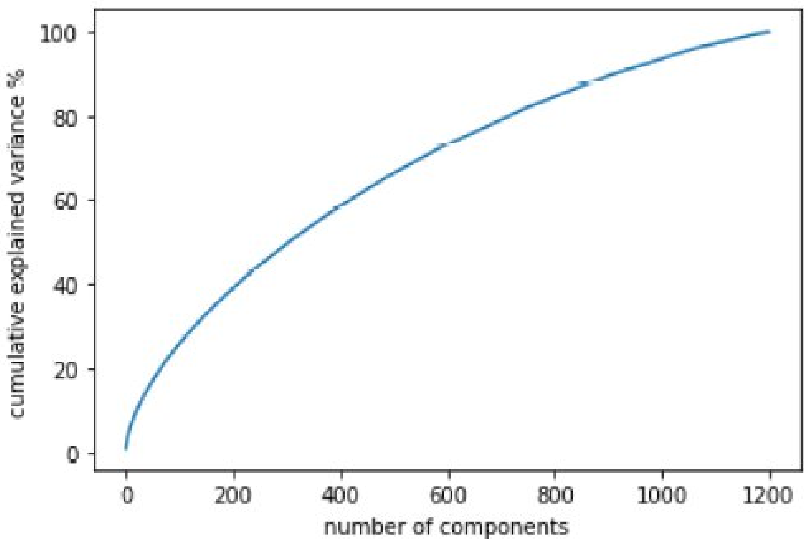


FIGURE 2.2: CEV as a function of components number for AlArabiya.

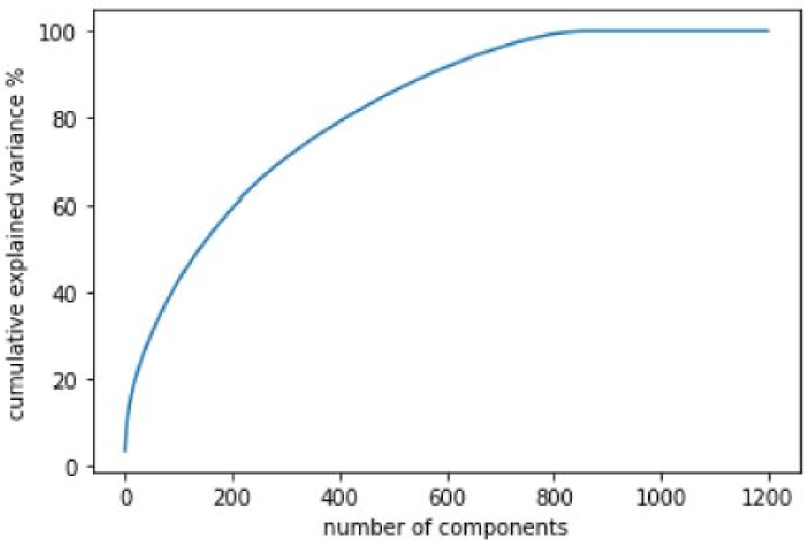


FIGURE 2.3: CEV as a function of components number for NADA.

2. **Classifiers performance based on CEV:**

   - **Use of Alarabiya Dataset:** Table 2.3 presents the classifiers' accuracy on AlArabiya dataset, corresponding to various CEV percentage values and the associated number of principal components (PCs), with the best performance for each classifier highlighted in bold. Based on the table, LR achieves the best accuracy of 99% when using 20% of the Cumulative Explained Variance (CEV). As the CEV increases, LR's accuracy decreases slightly, losing only 2% for larger CEV values. classifiers like LR, RF, and SVM maintain consistently high performance, with accuracy equal to or exceeding 97% across various CEV percentages.. In contrast, kNN experiences significant accuracy drops, performing well at 20% CEV (89.66%) but declining sharply with higher PCs. DT also fluctuates but stabilizes above 90% after 40% CEV. Overall, kNN is the most sensitive to the number of PCs, while LR, RF, and SVM are more robust across different CEV levels.

     It is important to note that all classifiers achieve their highest accuracy with a relatively small number of PCs (129 PCs or less). This is because the first PCs, which correspond to higher eigenvalues, capture the most general and significant features of the dataset. Conversely, a higher number of PCs, which have lower eigenvalues, tend to represent finer details rather than the general properties of a class, resulting in a decline in accuracy for all classifiers as more PCs are included.

TABLE 2.3: Classifiers' Accuracy Depending on CEV for Alarabiya.

Number of PCs	**CEV (%)**	**LR**	**kNN**	**DT**	**RF**	**SVM**
21	10	98.00	**92.33**	93.66	97.33	97.33
65	20	**99.00**	89.66	94.00	97.33	97.66
129	30	98.33	77.66	**94.66**	**98.33**	**98,33**
209	40	98.00	80.00	94.66	98.00	97.33
306	50	98.66	52.33	94.66	97.66	97.33
420	60	97.66	35.33	93.66	97.33	97.66
554	70	97.66	30.00	94.33	97.66	97.66
713	80	97.66	35.33	94.33	97.33	97.00
911	90	97.00	38.33	94.00	97.66	97.33
1156	99	97.33	20.00	93.00	97.66	97.00

- **Use of NADA Dataset:** Table 2.4 illustrates the classifiers' accuracy on AlArabiya dataset, corresponding to various CEV percentage values and the associated number of PCs, with the best performance for each classifier highlighted in bold. The table shows that LR, DT, RF, and SVM exhibit consistent performance across various levels of Cumulative Explained Variance (CEV), with LR achieving the highest accuracy of 99.00% at 60% CEV, although its performance slightly declines with higher CEV values. In contrast, kNN shows significant variability, performing worst with accuracy dropping to as low as 53.33% when 90% CEV is reached; it achieves better accuracy with fewer principal components (low CEV) but suffers when too many components are used. All classifiers require between 50 and 206 principal components to attain their optimal accuracy levels.

TABLE 2.4: Classifiers' Accuracy Depending on CEV for NADA.

Number of PCs	**CEV (%)**	**LR**	**kNN**	**DT**	**RF**	**SVM**
5	10	63.00	83.66	83.00	84.66	73.33
21	20	96.66	96.66	95.66	98.33	97.33
50	30	98.33	**97.00**	**97.33**	98.33	98.33
89	40	98.33	95.66	96.66	98.33	97.33
141	50	98.66	89.33	96.33	98.33	**98.66**
206	60	**99.00**	81.33	96.00	**98.66**	95.33
295	70	98.00	65.33	96.00	98.33	96.33
414	80	98.33	54.66	97.00	98.33	97.66
569	90	98.33	53.33	96.00	98.33	97.33
787	99	98.00	56.00	97.00	97.66	97.66

- **Overall Interpretation:** Overall, LR and RF demonstrate consistently high accuracy across different datasets and Cumulative Explained Variance (CEV) levels, highlighting their robustness and reliability. kNN shows variable performance, excelling at lower CEV levels but experiencing significant declines as CEV increases, indicating potential challenges with high-dimensional data. DT and SVM generally perform well, with SVM maintaining high accuracy across both datasets. Optimal performance for most classifiers typically occurs at lower to mid-range CEV levels (20%-60%). We can conclude that the number of principal components (PCs) required depends on both the underlying dataset and the specific classifier used.

3. **Projection onto the first two principal components (2D):** PCA reduces high-dimensional data to a lower-dimensional space for easier visualization. Projecting data onto the first two principal components (PCs) creates a 2D visualization, helping to reveal patterns, clusters, and relationships that may be obscured in higher dimensions. The projection of the AlArabiya and NADA datasets onto the first two PCs is shown in Figures 2.4 and 2.5, respectively. For the AlArabiya dataset, despite the complexity revealed by the Cumulative Explained Variance plot (Figure 2.2) , the PCA scatter plot demonstrates clear separation between the 5 classes in the first two PCs. This indicates that the first two components capture enough important information

for visual class separation, although more components are needed to fully capture the dataset's variance and underlying features. In contrast, the NADA dataset requires fewer components to explain most of the variance, as shown in the Cumulative Explained Variance plot (Figure 2.3). However, the PCA scatter plot shows that the first two components do not clearly separate the 10 classes. The overlap suggests that while fewer components capture the overall variance, they do not necessarily highlight the most class-distinguishing features, indicating that additional components are required for better class separation.

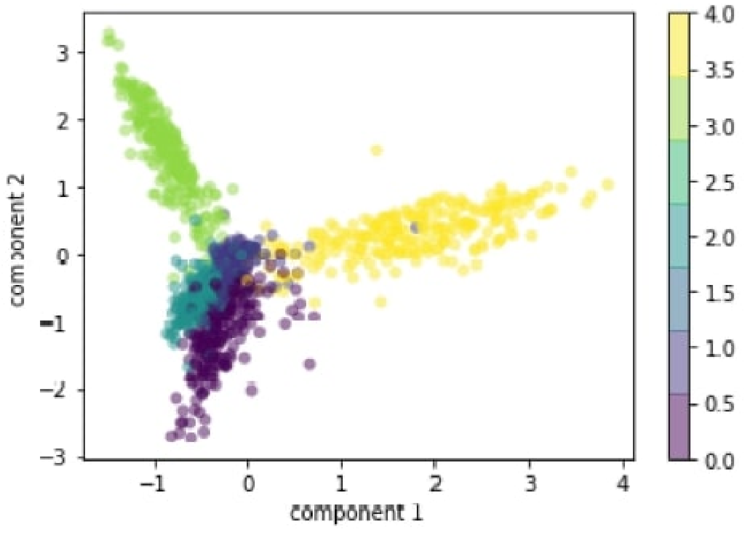


FIGURE 2.4: PCA Scatter Plot of AlArabiya Training Documents.

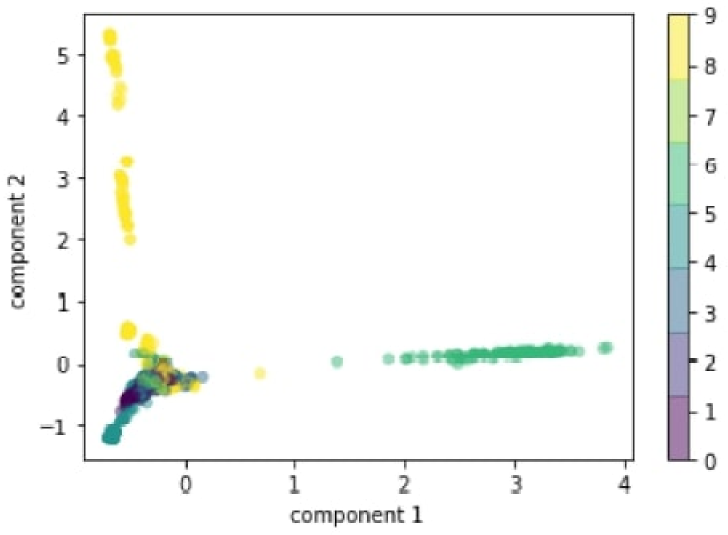


FIGURE 2.5: PCA Scatter Plot of NADA Training Documents.

4. **Effect of PCA usage on accuracy:**

TABLE 2.5: Accuracy of classifiers on Arabiya and NADA datasets with and without PCA

Classifier	**AlArabiya dataset**		**NADA dataset**	
	Accuracy Without PCA	**Accuracy With PCA**	**Accuracy Without PCA**	**Accuracy With PCA**
LR	97.33	**99.00**	97.33	**99.00**
KNN	**98.66**	92.33	97.33	97.00
DT	91.00	94.66	92.66	**97.33**
RF	98.33	98.33	**99.00**	98.66
SVM	98.00	98.33	96.33	**98.66**

Table 2.5 presents the accuracy of classifiers for both settings (with and without PCA) across both datasets, with the highest accuracy for each classifier highlighted in bold. The plot from Table is shown in Figure 2.6. The analysis shows that LR, DT, and SVM achieved their best results with PCA on the NADA corpus, while RF performed well on the NADA corpus without PCA. KNN uniquely achieved its best accuracy on the AlArabiya dataset without PCA. The lowest overall accuracy (91.00%) was recorded without PCA for DT on the AlArabiya corpus. Irrespective of the dataset,

PCA improved accuracy in 60% of cases, decreased it in 30%, and maintained it in 10%. The plot from Table 2.5 is shown in Figure 2.6 below.

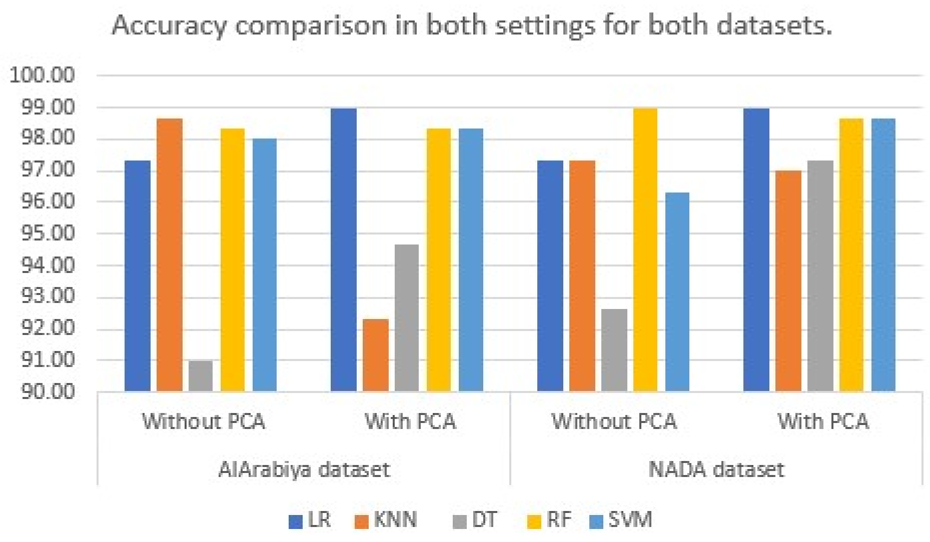


FIGURE 2.6: Accuracy comparison of the five classifiers under both settings (with PCA vs. without PCA)

5. **Effect of PCA usage on training time:**

TABLE 2.6: Training Time of Classifiers on Alarabiya and NADA Datasets with and without PCA.

Classifier	Arabiya dataset		NADA dataset	
	Training time(s) Without PCA	Training time With PCA	Training time Without PCA	Training time With PCA
LR	16.337	**0.001**	20.283	0.006
KNN	1655.972	**2.059**	1989.242	3.190
DT	25.078	1.392	31.489	**0.692**
RF	4.625	**0.960**	4.352	1.139
SVM	1690.165	**4.562**	1932.731	5.235

Table 2.6 shows the training times of classifiers in seconds for both scenarios (with and without PCA) across the two datasets, with the optimal training time for each classifier emphasized in bold. Given that both datasets consist of 1,200 training samples each, the minimum training times are achieved with PCA for LR, KNN, RF, and SVM on the AlArabiya dataset, while DT reaches its minimum training time with PCA on the

NADA dataset. Maximum training times are observed for various classifiers without PCA. Overall, PCA reduces training times for all classifiers across both datasets. The results from Table 2.6 are further illustrated in Figure 2.7 for clarity. Note that some training times for the non-PCA cases, such as 1690 seconds for LR, are beyond the range shown in Figure 2.7

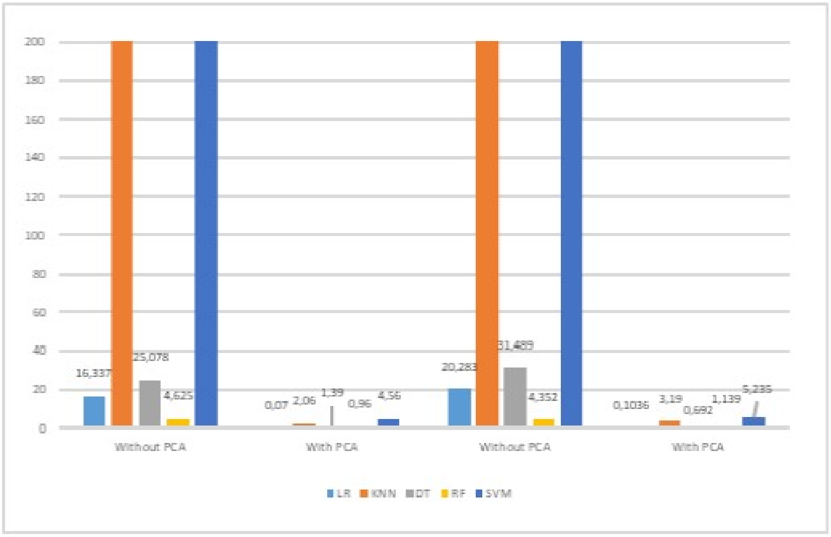
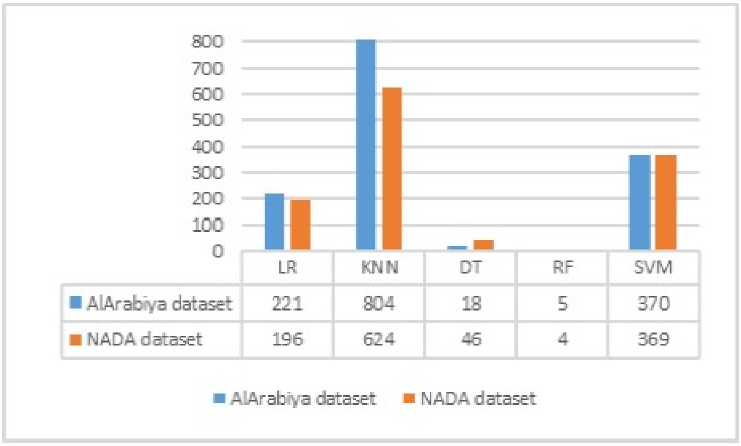


FIGURE 2.7: Training time of classifiers with and without PCA for both datasets.

6. **Time gain:** To summarize, we evaluate the time gain achieved by using PCA. As illustrated in Figure 2.8, the time gain ranges from 4 seconds (achieved by RF on the NADA dataset) to 804 seconds (achieved by KNN on the AlArabiya dataset), indicating that PCA accelerates classification between these two extremes.



	LR	KNN	DT	RF	SVM
AlArabiya dataset	221	804	18	5	370
NADA dataset	196	624	46	4	369

FIGURE 2.8: Gain in training time achieved with PCA.

2.4 Conclusion

In this chapter, we introduced a straightforward yet effective approach to address the challenge of high-dimensional feature spaces in text classification by applying PCA for feature extraction in Arabic text classification. We evaluated PCA's impact on five popular classifiers, including LR, KNN, DT, RF, and SVM. The results showed significant improvements in classification accuracy for most classifiers, with enhancements observed in 60% of the cases. Additionally, PCA notably reduced training time, achieving reductions of up to 800-fold.

The next chapter presents the second contribution to Arabic text classification, which involves a distance-based meta-features approach. This method differs from the first contribution in several aspects, requiring fewer dimensions and lower computational requirements.

Chapter 3

Distance-Based Meta-Features for Arabic Text Classification

3.1 Introduction

The TF-IDF method faces challenges such as high dimensionality and data sparsity, which result in large storage demands, increased computational costs, and a heightened risk of overfitting, ultimately degrading the performance of classification systems. To overcome these issues, various studies have utilized meta-features derived from clustering methods (Kyriakopoulou and Kalamboukis, 2007), the kNN method (Canuto, Gonçalves, and Benevenuto, 2016), and category centroids (Pang, Jin, and Jiang, 2015). Furthermore, distance-based meta-features have demonstrated considerable effectiveness in text classification, as shown in (Canuto et al., 2018; Cunha et al., 2020). In this chapter, we introduce an additional preprocessing step to the classification pipeline by generating distance-based meta-features derived from the original TF-IDF representations. Specifically, we focus on four types of distance-based meta-features: CosKNN, L2KNN, CosCent, and L2Cent, and evaluate their their effectiveness and efficiency as a dimensionality reduction technique for Arabic text classification. The impact of these features is assessed using four widely-used classifiers: k-Nearest Neighbors, Logistic Regression, Random Forest , and Support Vector Machine. To the best of our knowledge, this represents the first application of distance-based meta-features in the context of Arabic text classification.

The rest of the chapter is organized as follows: Section 3.2 offers an overview of the related works. Section 3.3 outlines the proposed methodology utilized in this study. Section 3.4 details the experimental setup, while Section 3.5 presents and discusses the obtained results. Lastly, Section 3.6 concludes the chapter.

3.2 Related works

In the literature, numerous feature extraction and representation methods based on meta-features have been proposed for classifying textual data (Kyriakopoulou and Kalamboukis, 2007; Canuto, Gonçalves, and Benevenuto, 2016; Pang, Jin, and Jiang, 2015; Canuto et al., 2018; Cunha et al., 2020) and images (Tsai et al., 2011).

In the work presented in (Canuto, Gonçalves, and Benevenuto, 2016), the authors introduced new meta-level features for sentiment analysis, specifically targeting short messages. They conducted experiments comparing these features to the original bag-of-words approach, previously proposed state-of-the-art meta-features in the literature, lexicon-based sentiment analysis methods, and supervised ensembles of lexicon-based techniques. The proposed feature set has the potential to transform the original feature space into a smaller, yet more informative space. Experiments conducted with these new meta-features on nineteen benchmark datasets demonstrated significant improvements in effectiveness across most datasets compared to all baselines. In fact, the proposed approach outperformed all others in every tested dataset and scenario.

The authors in (Cunha et al., 2020) proposed three additional preprocessing steps in the text classification system to enhance effectiveness while minimizing associated costs, addressing the effectiveness versus efficiency trade-off. The first step, distance-based meta-features generation, aims to reduce the feature space size while potentially creating a more informative representation. The second step, sparsification, seeks to increase data sparsity, and the third step, selective sampling, focuses on reducing the number of instances. Experiments were conducted on four textual datasets: the WebKB dataset, which consists of 8,199 documents organized into 7 classes; the 20NewsGroup dataset, containing 18,846 documents divided into 20 newsgroups; the ACM dataset, with 24,897 articles categorized into 11 classes; and the classical Reuters dataset, comprising 13,327 news articles organized into 90 categories. The results indicated that these three proposed preprocessing steps led to substantial improvements in effectiveness compared to TF-IDF and embedding-based representations, along with reductions in execution time.

In the study described in (Gopal and Yang, 2010), the authors proposed a novel approach for multi-label text classification by introducing meta-level features that effectively represent the relationships between each instance and multiple classes. This method transforms the classical representation of instances (TF-IDF) and categories into a meta-level feature space. Experiments were conducted on six benchmark datasets, including Emotion, Scene, Yeast, Citeseer, Reuters21578, and Vowel, using eight performance metrics. The results demonstrated significant performance improvements over previous state-of-the-art methods, including Rank-SVM (Elisseeff and Weston, 2001), ML-KNN (Zhang and Zhou, 2007), and

IBLR-ML (Cheng and Hüllermeier, 2009).

To the best of our knowledge, distance-based meta-features have not yet been applied to Arabic text classification. This gap presents a unique research opportunity, as integrating these meta-features could enhance the understanding of relationships within the data and improve both the effectiveness and efficiency of classification.

3.3 Proposed Methodology

This section outlines the methodology for implementing Distance-Based Meta-Feature in Arabic text classification. The approach encompasses several critical phases, as illustrated in Figure 3.1. It begins with several preprocessing steps, including tokenization, cleaning, and stemming, followed by encoding the text into numerical representations. The documents are then transformed using dimensionality reduction techniques through meta-feature generation. These reduced features are subsequently fed into the classifier to generate predictions.

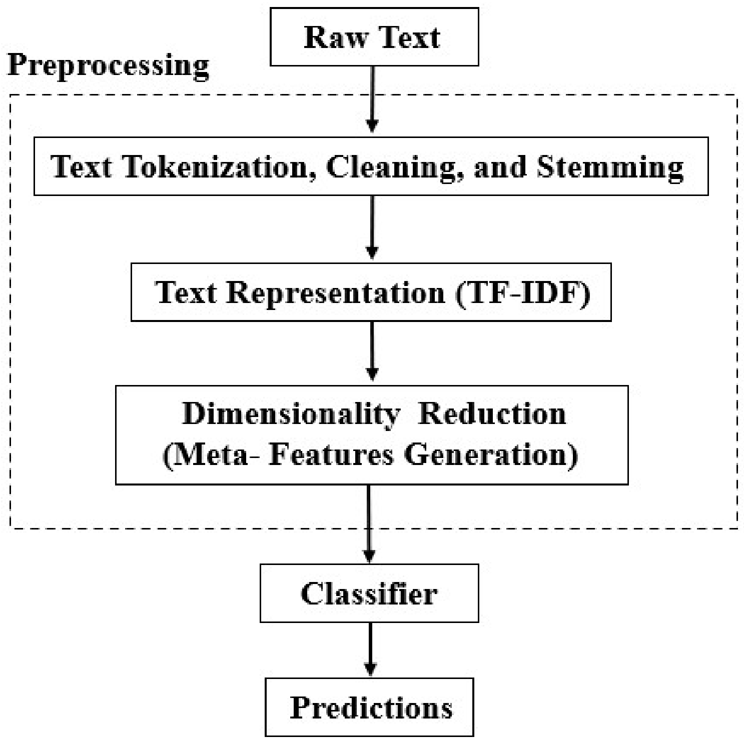

Raw Text
Preprocessing
Text Tokenization, Cleaning, and Stemming
Text Representation (TF-IDF)
Dimensionality Reduction (Meta- Features Generation)
Classifier
Predictions

FIGURE 3.1: The proposed Arabic text classification pipeline.

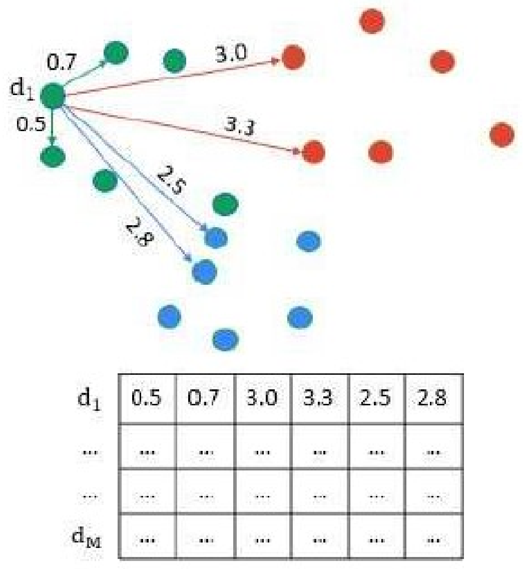
3.3.1 Preprocessing

Effective preprocessing is vital for the success of any text classification task, especially when dealing with the Arabic language, which has unique characteristics. However, there is currently no standardized preprocessing method specifically for Arabic text classification. In this study, we applied the same traditional preprocessing steps outlined in Chapter 2, Subsection Subsection 2.2.3, before generating Meta-Features. These steps included text cleaning, which involved removing digits, special characters, non-Arabic words, and Arabic stop words; tokenization, where the texts were split into words using the Camel_tools tokenizer; stemming, performed with the ISRI stemmer ; and text representation, for which TF IDF was chosen as the encoding method.

3.3.2 Meta-features generation

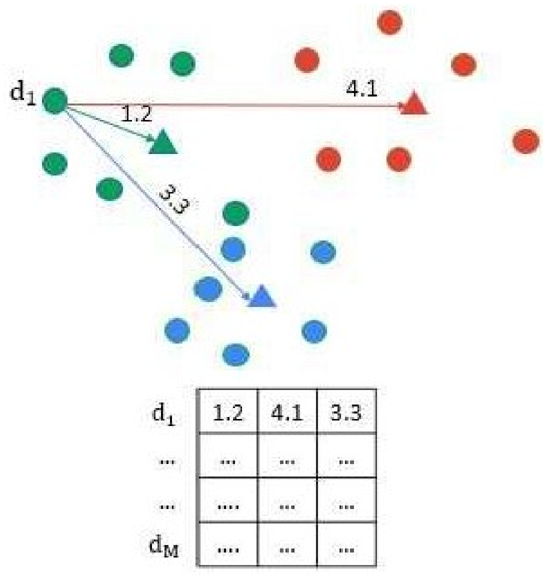
Meta-features are attributes that capture relationships between textual documents in a dataset. Unlike traditional features that represent isolated words or phrases, meta-features provide deeper insights by summarizing interactions between documents. These additional attributes can significantly enhance the performance of machine learning algorithms in text classification, especially in high-dimensional spaces where conventional text features, such as word counts or TF-IDF, may be sparse or inadequate. Feeding meta-features into classifiers allows them to focus more effectively on key patterns and relationships between documents, potentially enhancing classification performence.

In this study, we focus on distance-based meta-features (DBMFs), as they have been shown to produce significant results (e.g., as demonstrated in (Canuto et al., 2018). Generating DBMFs is an additional step in the preprocessing phase aimed at reducing the dimensionality and sparsity of the TF-IDF representation. Instead of representing the entire document, DBMFs capture only the distances between a document and other labeled documents or the class centers. We consider two types of DBMFs: kNN meta-features and centroid meta-features. kNN meta-features are derived by selecting the k nearest documents from each class relative to a given document, while centroid meta-features are generated by calculating the distance between a document and the center of each class. Distances for both kNN and centroid meta-features are computed using either Euclidean distance (L2KNN, L2Cent) or cosine similarity (CosKNN, CosCent). Figures 3.2 and 3.3 illustrate the process of generating kNN and centroid meta-features using Euclidean distance. In these examples, colored circles represent documents and their corresponding three classes, while colored triangles represent the class centroids. To generate kNN meta-features for a specific document (d1), the k nearest documents from the green class are selected (in this case, k=2), yielding two meta-features (0.5 and 0.7). This process is repeated for the red and blue classes, resulting in a vector assigned to d1: (0.5, 0.7, 3.0, 3.3, 2.5, 2.8), where each value represents a kNN meta-feature. This procedure is then repeated for all remaining documents in both the training and test sets. For centroid meta-features, the distance between a document (d1) and each class center is calculated. The resulting vector for d1 is represented as (1.2, 4.1, 3.3), where each value corresponds to a centroid meta-feature. Similar to the kNN meta-features, this calculation is performed for all documents in the training and test sets.



d_1	0.5	0.7	3.0	3.3	2.5	2.8
...	...	...	...	...	...	...
...	...	...	...	...	...	...
d_M	...	...	...	...	...	...

FIGURE 3.2: Example of kNN meta-features generation.



d_1	1.2	4.1	3.3
...	...	...	...
...	....	...	...
d_M	....	...	...

FIGURE 3.3: Example of centroid meta-features generation.

Table 3.1 summarizes the distance-based meta-features utilized in this study, along with their dimensionality and corresponding references. Here, C denotes the number of classes and k the number of neighbors.

TABLE 3.1: Distance-Based Meta-Features

Meta-Feature	Dimensions	Reference
CosKNN	C × K	(Cunha et al., 2020; Gopal and Yang, 2010)
L2KNN	C × K	(Cunha et al., 2020; Gopal and Yang, 2010)
CosCent	C	(Cunha et al., 2020; Gopal and Yang, 2010; Pang, Jin, and Jiang, 2015)
L2Cent	C	(Cunha et al., 2020; Gopal and Yang, 2010)

3.4 Experimental Setup

3.4.1 Dataset

In our experiments, we used the same dataset as in (Louail, Kara-Mohamed Hamdi-Cherif, and Hamdi-Cherif, 2021), a balanced collection of 1,500 Arabic text documents suitable for Arabic text classification tasks. The documents are categorized into five domains: medicine, sports, politics, technology, and finance. The dataset was split into 80% for training and 20% for testing.

3.4.2 Hyperparameter tuning

The optimal parameters were identified through a grid search, where a predefined set of parameters was evaluated using 5-fold cross-validation. This process aimed to determine the optimal number of neighbors k and the best weight function for the kNN algorithm, the the best regularization parameter C for the SVM with a linear kernel, and the ideal number of trees and split quality criterion for the Random Forest classifier.

The number of neighbors k used for generating the kNN meta-features was chosen from a set of five values: {5, 9, 13, 17, 21}. Through this process, the optimal value was identified as k = 9. Table 3.2 summarizes the hyperparameters used for tuning various classifiers. For Logistic Regression, the penalty is set to l2 with a regularization strength C of 1. The K-Nearest Neighbors (kNN) classifier tests different numbers of neighbors (ranging from 17 to 35) and uses either uniform or distance-based weighting. The Random Forest classifier evaluates two configurations, using either 50 or 100 trees, with the Gini impurity or entropy criterion to assess split quality. The Support Vector Machine classifier employs a linear kernel and tests various regularization strengths C from 0.001 to 100.

TABLE 3.2: Tuning hyperparameters for classifiers.

Classifier	**H**yperparameter

LR	Penalty=l2
	C=1

KNN	n_neighbors=[17,19,21,23,25,27,29,31,33,35]
	weights=['uniform','distance']

RF	n_estimators=[50,100]
	criterion=['gini','entropy']

SVM	kernal='linear'
	C=[0.001,0.01,0.1,1,10,100]

3.5 Results and discussions

3.5.1 Dimensionality reduction using Meta-Features

Table 3.3 illustrates a substantial reduction in the number of dimensions when transitioning from the original TF-IDF representation. In this context, 5 represents the number of classes, while 9 is the optimal number of neighbors for both CosKNN and L2KNN. Specifically, there is a decrease of 99.85% from 29,300 dimensions to 45 dimensions for both CosKNN and L2KNN. Furthermore, the dimensionality is reduced by 99.98% for CosCent and L2Cent, which drop to only 5 dimensions. This indicates that CosKNN and L2KNN retain a higher number of dimensions due to their dependence on both the number of classes and the value of k. In contrast, the dimensionality of CosCent and L2Cent is solely influenced by the number of classes, resulting in a much more compact representation.

TABLE 3.3: Representation Methods and Their Number of Dimensions

Representation Method	**Dimensions**
TF-IDF	29,300
CosKNN	$5 \times 9 = 45$
L2KNN	$5 \times 9 = 45$
CosCent	5
L2Cent	5

3.5.2 Classifiers' accuracy

After training our classifiers on the TF-IDF baseline representation and the different DBMFs groups using the optimal hyperparameters, we evaluated the impact of DBMFs groups on classifiers' accuracy, expressed as percentages, as shown in Table 3.4. The best values for each classifier are highlighted in bold. Several key observations can be made from this table:

- LR and SVM achieved their highest accuracy using meta-features, showcasing their strength in leveraging such feature sets.
- KNN and RF performed best with TF-IDF, but they also showed strong results with meta-features. Notably, there is only a 0.33% decrease in accuracy for both classifiers when using meta-features.
- SVM is the most consistent and high-performing classifier, with its performance peaking at 99.00% when using the CosCent method.
- SVM emerged as the best classifier with the DBMFs groups, demonstrating its robustness in handling both local information (through KNN-based features) and global information (via centroid-based features). Notably, the highest accuracy, 99.00%, was achieved by SVM using the CosCent method.
- The L2Cent method achieved the highest overall accuracy average across all classifiers.

In conclusion, accuracy improved in 50% of the cases, while in the remaining cases, the decrease in accuracy was minimal, with only a 0.33% drop.

TABLE 3.4: Classifiers' Accuracy Comparison with TF-IDF and DBMFs Groups.

Classifier	TF-IDF	L2KNN	CosKNN	L2Cent	CosCent
LR	97.33	**98.33**	**98.33**	**98.33**	97.66
KNN	**98.66**	98.00	98.33	98.33	98.33
RF	**98.33**	98.00	97.00	97.66	97.33
SVM	98.00	98.33	98.33	98.66	**99.00**
Average	98.08	98.17	97.99	**98.25**	98.08

Figure 3.4 below presents a visual representation of the data from Table 3.4, providing a clearer comparison of the classifiers' performance across different feature sets.

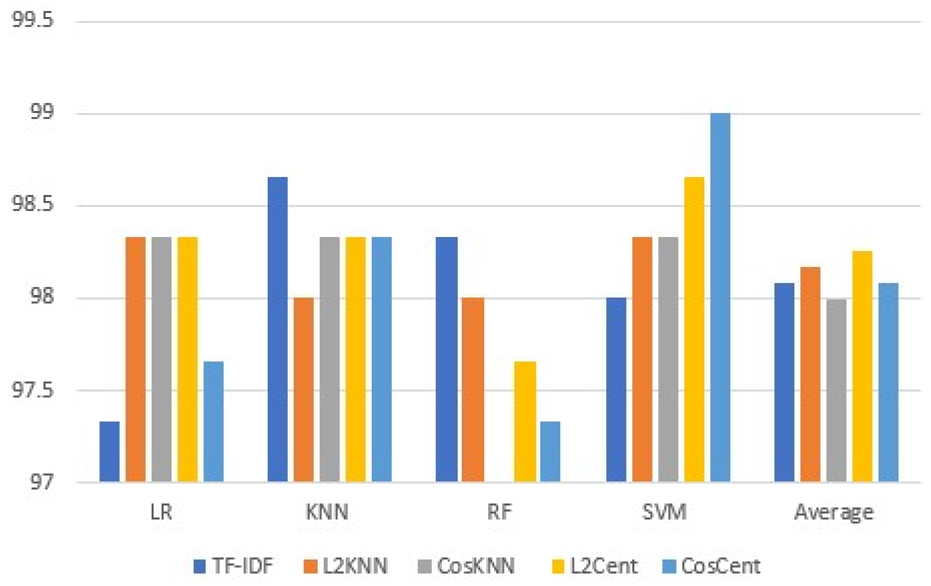


FIGURE 3.4: Classifiers' Accuracy Comparison with TF-IDF and DBMFs Groups.

3.5.3 Training time

Table 3.5 shows the training times (in seconds) for different classifiers across the various distance-based meta-feature representations and the TF-IDF baseline.The best training times are highlighted in bold. Based on Table 3.5, it is evident that:

- The highest training times were recorded for different classifiers when using the TF-IDF representation, as the classifiers are fed with a large number of dimensions.
- The lowest training times were observed when using meta-features, due to the smaller number of dimensions in the meta-feature space.
- The minimum training time for all classifiers was achieved with L2Cent.
- The best average training time was obtained with centroid meta-features (i.e., L2Cent and CosCent), as they have the smallest number of features.

TABLE 3.5: Classifiers' Training Times (s) with TF-IDF and DBMFs Groups.

Classifier	TF-IDF	L2KNN	CosKNN	L2Cent	CosCent
LR	16.33	2.01	0.29	**0.07**	0.67
KNN	1655.97	4.60	4.67	**1.28**	2.62
RF	4.62	0.43	0.49	**0.25**	0.28
SVM	1690.16	1.04	2.54	**0.52**	0.95
Average	841.77	2.03	2.00	**0.53**	1.14

3.5.4 Time gain

Table 3.6 illustrates the effect of distance-based meta-features on the speed of the Arabic text classification process. For better clarity, Figure 4 presents a graphical representation of the data from Table 3.6.

TABLE 3.6: Training Time Gain with DBMFs.

Classifier	L2KNN	CosKNN	L2Cent	CosCent
LR	8	56	**224**	24
KNN	359	354	**1287**	630
RF	11	9	**18**	16
SVM	1613	664	**3244**	1764

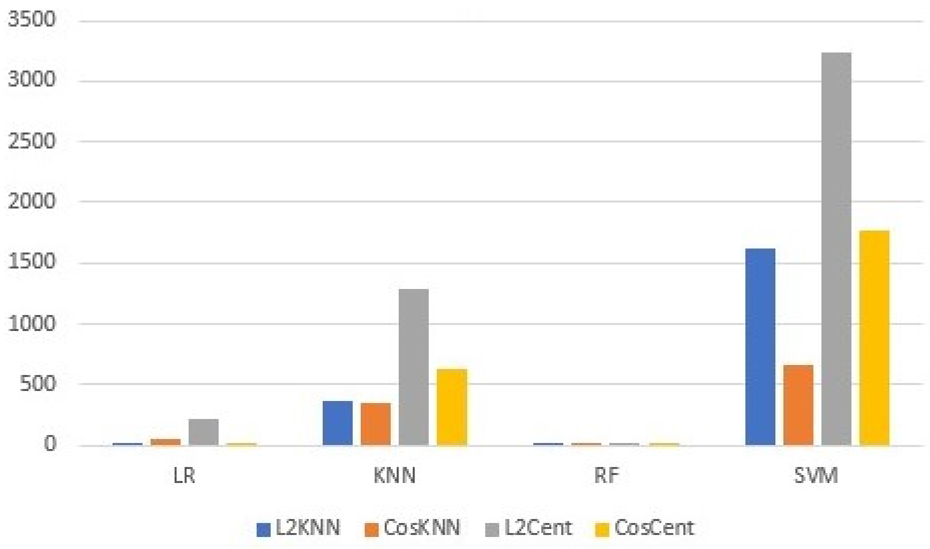


FIGURE 3.5: Time Gain with DBMFs Across Different Classifiers.

From Figure 3.5 , we can observe the following:

- The use of meta-features led to significant speedup gains, with improvements ranging from a minimum of 8 times (achieved by LR with L2KNN) to a maximum of 3244 times (achieved by SVM with L2Cent).
- SVM achieved the highest time gain across all meta-features, followed by the kNN classifier.

3.5.5 Comparing DBMFs with PCA

In this study, we employed Principal Component Analysis (PCA) to reduce the dimensionality of the TF-IDF representation. We chose to retain principal components that accounted for 95% of the cumulative variance, reducing the dimensionality from the original 29,298 dimensions to a more compact 911 dimensions. Table 3.7 provides an overview of the accuracy performance achieved by applying PCA and meta-features to reduce the dimensionality of the data. Based on this table, all classifiers demonstrated improved performance with the integration of meta-features. Notably, the kNN classifier combined with PCA yielded the lowest accuracy among the classifiers. For enhanced clarity and visualization, Figure 3.6 illustrates the data presented in Table 3.7.

TABLE 3.7: Comparison of Classifiers' Accuracy Using Meta-Features vs. PCA.

Classifier	TF-IDF	L2KNN	CosKNN	L2Cent	CosCent	PCA
LR	97.33	**98.33**	**98.33**	**98.33**	97.66	97.33
KNN	**98.66**	98.00	98.33	98.33	98.33	39.33
RF	**98.33**	98.00	97.00	97.66	97.33	97.33
SVM	98.00	98.33	98.33	98.66	**99.00**	97.00

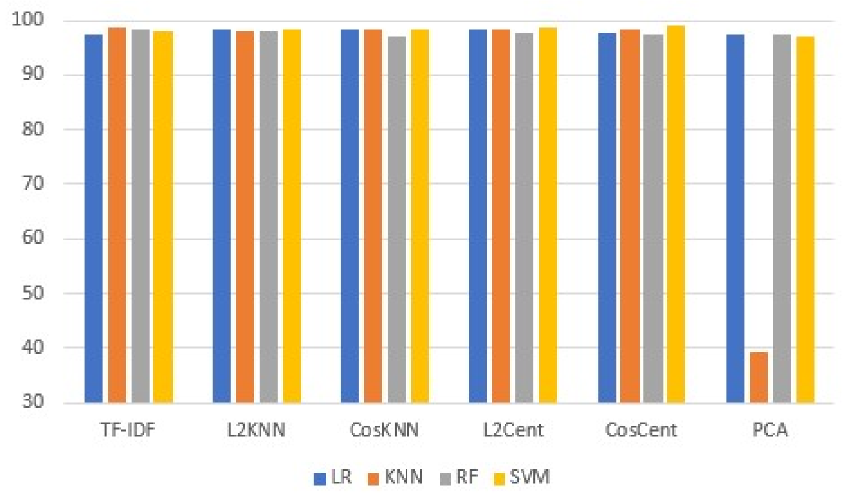


FIGURE 3.6: Comparison of Classifiers' Accuracy Using Meta-Features vs. PCA..

3.5.6 Statistical evaluation

We conducted a paired t-test to evaluate the significance of the accuracy improvements achieved with our proposed method. The paired t-test is a statistical tool used to compare the means of paired observations within each group. This test was performed to determine whether there was a significant difference between the means of the paired observations. The null hypothesis assumes no significant difference between the means. If the resulting p-value is below a predetermined significance level (e.g., 0.05), the null hypothesis is rejected, indicating that the observed performance difference is statistically significant. To generate the population for a paired t-test in the context of text classification, we created five randomizations of the training set, resulting in multiple pairs of model performance measurements for each realization. Each pair comprised the accuracy obtained using the distance-based meta-features representation and the accuracy obtained using the TF-IDF representation. Both representations were evaluated on the same shuffled training set using the same classifier (SVM). Table 3.8 presents the results of the paired t-test, comparing the performance of the best meta-features with lower time costs (L2Cent, CosCent) against the TF-IDF representation.

TABLE 3.8: P-Values of the Paired T-Test.

Method	**CentL2**	**CosCent**
TF-IDF	0.01	0.03

From Table 3.8, we observe that all obtained p-values are below 0.05 within a 95% confidence interval, clearly indicating the rejection of the null hypothesis in each case. This signifies a significant improvement in accuracy achieved by L2Cent and CosCent, suggesting that the observed results are not due to random chance.

3.6 Conclusion

This chapter emphasizes the importance of prioritizing the pre-processing phase over classifier algorithms to improve text classification performance while reducing time-related costs. We introduced a new pre-processing step for Arabic texts to tackle the issues of high dimensionality and sparsity associated with TF-IDF representation, which can negatively affect classification performance. We used four distance-based meta-features , namely CosKNN, L2KNN, CosCent, and L2Cent as dimensionality reduction methods and evaluated their impact on KNN, SVM, RF, and LR classifiers. Results showed improved classification accuracy in 50% of cases, along with a significant reduction in training time, up to 3244 times faster for all classifiers. This marks the first use of distance-based meta-features for Arabic text classification.

The next chapter will focus on testing our approach with larger datasets and examining the benefits of integrating local and global information by combining KNN-based and centroid-based meta-features with word embeddings to further enhance performance.

Chapter 4

Tasneef: A Fast and Effective Hybrid Representation Approach for Arabic Text Classification

4.1 Introduction

Many works in ATC rely on a single text representation method, such as TF-IDF, which simplifies the modeling process but may fail to capture the diverse linguistic aspects of Arabic. This approach often leads to incomplete semantic understanding and can result in high time and memory consumption, as demonstrated by experiments. While deep learning-based representations can be effective, they also come with high computational costs, particularly affecting real-time applications. To address these challenges, this chapter presents *Tasneef* [1], a novel hybrid approach aimed at tackling computational challenges in ATC by specifically reducing memory consumption and runtime overhead. Two key issues are tackled: first, analyzing the statistical properties of text using TF-IDF, enhanced through dimensionality reduction with Distance-Based Meta Features (DBMFs), which mitigate the high dimensionality and sparsity issues of TF-IDF. Second, semantic features are captured using fastText word embeddings. The chapter emphasizes that relying on a single representation method, such as TF-IDF, often results in high computational costs and may fail to capture critical information, particularly in complex languages like Arabic. By integrating multiple representations, the goal is to optimize ATC performance while minimizing computational resources, making the approach suitable for mid-sized environments or resource-constrained research groups.

The chapter is organized as follows: Section 4.2 details the architecture of the proposed method, explaining the various modules and their interactions. Section 4.3 describes the experimental setup and the main tools used. Section 4.4 discusses the results, highlighting

[1] *Tasneef* is the Arabic word for classification.

the strengths and weaknesses of the proposed method. Finally, Section 4.5 concludes the chapter and suggests potential future developments to address its limitations.

4.2 Methodology

4.2.1 Overall architecture

- **Proposed Pipeline:** The structure of *Tasneef* is depicted in Figure 4.1. The pipeline involves several key steps. Starting with a raw dataset, a preprocessing procedure is applied on the text to clean and structure it. On the statistical level, we employ the TF-IDF for encoding the text, as a sparse vector. Afterward, we reducethe TF-IDF feature space by generating DBMFs, resulting in a dense vector. We consider two types of reduction: local, using kNNL2 and kNNCos; and global, based on CentL2 and CentCos. The resulting hybrid dense representation is then obtained by concatenating the DBMFs with the fastText embeddings. Finally, the concatenated features are fed into the SVM for final classification.

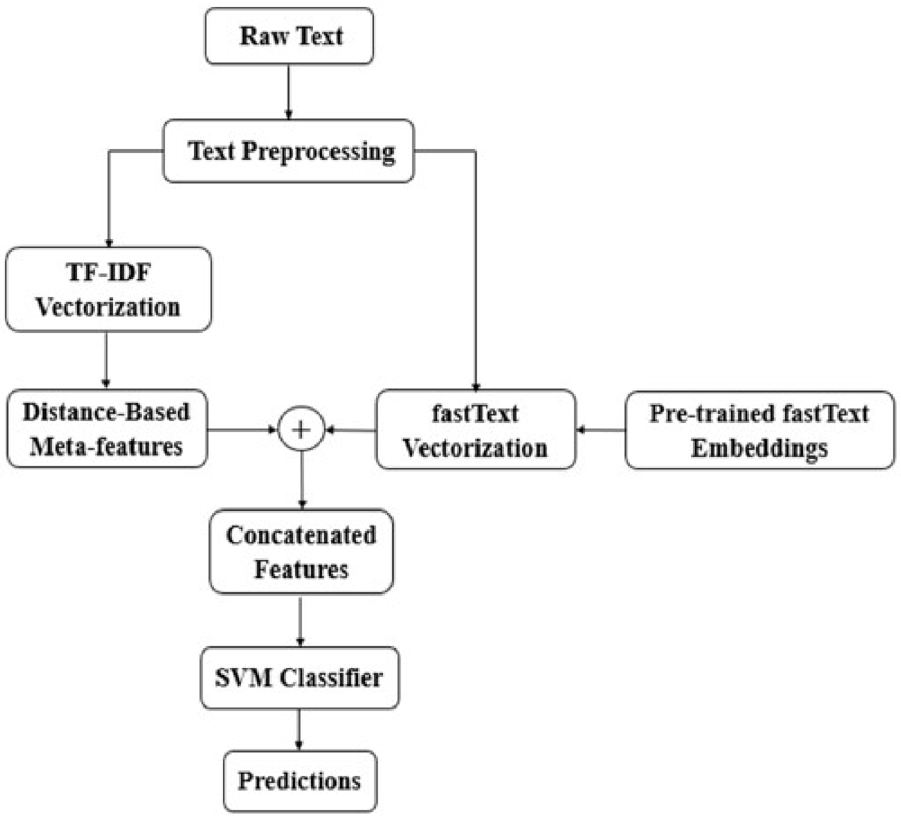


FIGURE 4.1: *Tasneef* architecture.

- **Main algorithm:** Algorithm 1 outlines the phases proposed by *Tasneef*, further expanded in the following subsections.

4.2.2 *Tasneef* Text preprocessing

For all datasets, we conducted preprocessing operations that align with widely adopted practices in the field ofATC, which have consistently yielded the most reliable results. Traditional practices include:

- **Data cleaning**: We start with a raw benchmark dataset and pre-process the text by removing numbers, special characters, non-Arabic words, and Arabic stop-words. We employed the most extensive Arabic stop-words list available [2], which comprises 13,629 stop-words.

- **Tokenization**: After conducting multiple trials, we chose the PyArabic [3] tokenizer (Zerrouki, 2023) over the Camel_tools [4] tokenizer (Obeid et al., 2020) because of its faster performance, particularly with large datasets, while retaining an equivalent level of effectiveness.

- **Stemming:** Although lemmatization is more accurate in reducing words to their base or root forms, it is computationally more expensive and requires advanced linguistic resources. In contrast, stemming is faster and simpler, making it more suitable for tasks where speed is essential. Despite its reduced accuracy, stemming is often sufficient for document classification and other NLP tasks. In this case, the ISRI Arabic stemmer from the NLTK library was chosen for its versatility and popularity.

- **Normalization:** In our study, we opted not to normalize Arabic letters to prevent altering the contextual meaning of particular words, diverging from certain alternative methodologies. For example, the word (شأنٌ, matter, affair, position) can be confused with (شانَ, to tarnish), the word (أمالَ, to curve, to bend) with (آمالٌ, hopes), and the word (آلهة, divinities) with (ألهه, distracted him, or distract him!). These simple examples demonstrate how normalization can have a significant negative impact on word meaning.

[2]https://www.nltk.org/
[3]https://pyarabic.readthedocs.io/ar/latest/features.html
[4]https://camel-tools.readthedocs.io/en/latest/

4.2.3 Statistical property and DBMFs construction in *Tasneef*

Dimension reduction enhances computational efficiency by decreasing the feature space. After vectorization with TF-IDF, DBMFs are used to reduce vector dimensions. We describe how DMBFs are calculated.

4.2.3.1 DBMFs distance calculation

In *Tasneef*, we designed DBMFs that combine both local information, achieved through kNN, and global information, handled through centroid (Cunha et al., 2020). Distances are calculated for both kNN and centroid DBMFs by using Euclidean distance or cosine distance, as explained in Equations 4.1, 4.2, and 4.3 below:

- *Euclidean distance (L2)*: in n dimensional Euclidean space, the Euclidean distance d between two points $\mathbf{p}$ and $\mathbf{q}$ is given by:

$$d = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{4.1}$$

  where p_i (or q_i) is the i-th coordinate of $\mathbf{p}$ (or $\mathbf{q}$).

- *The cosine similarity* $\cos(\theta)$ between two vectors $\mathbf{a}$ and $\mathbf{b}$ is the result of their dot product divided by the product of their respective lengths, and is calculated as:

$$\cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}|| \cdot ||\mathbf{b}||} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \cdot \sqrt{\sum_{i=1}^{n} b_i^2}} \tag{4.2}$$

  where a_i (or b_i) is the i-th coordinate of $\mathbf{a}$ (or $\mathbf{b}$).

- *The cosine distance* is then defined as:

$$\text{cosine\_distance}(\mathbf{a}, \mathbf{b}) = 1 - \cos(\theta) \tag{4.3}$$

  As a result, the cosine distance metric measures the angle between two vectors and ranges from 0 (if the vectors are identical) to 1 (if the vectors are orthogonal).

4.2.3.2 Local DBMFs obtainment

In the phase of local processing, we rely on kNN DBMFs, which involve selecting the k nearest documents from each class (e.g., Politics, Culture, etc.) to a specific document. In this context, our method incorporates two key DBMFs. Firstly, kNNL2 utilizes the L2 (or Euclidean) distance to identify the k nearest documents from each class based on their Euclidean distances, described by Equation 4.1. Additionally, kNNCos employs the cosine similarity metric to compute the kNN DBMFs, selecting the k nearest documents from each class to a given document based on their cosine similarity scores, based on Equation 4.3. Algorithm 2 describes the steps involved in local DBMFs generation.

4.2.3.3 Global DBMFs obtainment

In the global feature processing phase, we employ centroid DBMFs to assess the relationship between documents and the center of each class. This involves two distinct approaches: CentL2 and CentCos. CentL2 computes the centroid DBMFs using the L2 (Euclidean) distance metric, quantifying the distance between a document and the centroid (mean vector) of each class based on their Euclidean distances. On the other hand, CentCos utilizes the cosine similarity metric to compute the centroid DBMFs, measuring the similarity between a document and the centroid of each class based on their cosine similarity scores. Equations 4.1, 4.2, and 4.3 still hold here, with the centroid considered as a point. Algorithm 3 describes the steps involved in global DBMFs generation.

4.2.3.4 Resulting DBMFs

The steps outlined above allow us to generate the DBMFs as presented in Table 4.1. A brief description of each DBMFs group is given along with its number of dimensions, where C represents the number of classes and k represents the number of neighbors used to generate kNN DBMFs. In terms of notation, ‘Cent’ means the combination of both the CentL2 and CentCos as centroid-based DBMFs groups.

Algorithm 2 Local DBMFs - kNN DBMFs Generation From TF-IDF Feature Matrix

```
Input:
    X_train: TF-IDF feature matrix for the training set
    X_test: TF-IDF feature matrix for the test set
    Y_train: Training set class labels
    k: Number of nearest neighbors
Output:
    kNN_MFs_train: kNN-based MFs for the training set
    kNN_MFs_test: kNN-based MFs for the test set
Begin:
    // Function to generate kNN MFs//
    // Refer to Equ. 4.1, 4.2 & 4.3 for distance calculation//
    1. Function Generate_kNN_MetaFeatures(X, X_train, Y_train, k):
      a. Initialize Meta_Features as an empty matrix with dimensions
      $[\text{length}(X), \text{length}(\text{unique}(\text{Y\_train})) \times k]$
      b. For i = 1 to length(X):
          Initialize MFs_vec as an empty list
          i. For c = 1 to length (unique(Y_train)) do:
              - Docs_c ={documents in X_train where Y_train == c}
              If X[i] belongs to X_train:
                  - Docs_c = Docs_c excluding X[i]
              EndIf
              - Distances = Calculate_distances (X[i], Docs_c)
              -Nearest_neighbors=Select_k_nearest_neighbors(Distances,k)
              -Append Distances of Nearest_neighbors to MFs_vec
          EndFor
          ii. Store MFs_vec in Meta_Features for document i
              - Meta_Features[i] = MFs_vec
      EndFor
      c. Return Meta_Features
    End Function
    // Generate kNNbased MFs for the training set//
    2.kNN_MFs_train=Generate_kNN_ MetaFeatures(X_train, X_train, Y_train, k)
    //Generate kNNbased MFs for the test set //
    3.kNN_MFs_test=Generate_kNN_MetaFeatures(X_test, X_train, Y_train, k)
End
```

Algorithm 3 Global DBMFs - Centroid DBMFs Generation From TF-IDF Feature Matrix

```
Input:
    X_train: TF-IDF feature matrix for the training set
    X_test: TF-IDF feature matrix for the test set
    Y_train: Training set class labels
Output:
    Centroid_MFs_train: Centroid-based MFs for the training set
    Centroid_MFs_test: Centroid-based MFs for the test set
Begin
    // Step 1: Centroid calculation //
    1. For each class c in unique (Y_train):
        a. Compute the centroid of class c in X_train
            - Centroid_c = mean( TF-IDF vectors of all training documents in class c)
        b. Store Centroid_c in Centroids
    EndFor
    // Step 2: Function for generating centroid MFs //
    // Refer to Equ. 4.1, 4.2 & 4.3 for distance calculation //
    2. Function GenerateMetaFeatures (X, Centroids):
        a. Initialize an empty matrix Meta_Features with dimensions
        [length(X) x length (Centroids)]
        b. For i = 1 to length(X):
            i. For c = 1 to length (Centroids):
                - Compute the distance between X[i] and Centroids[c]
                - Store the distance in Meta_Features[i, c]
            EndFor
        EndFor
        c. Return Meta_Features
    End Function
    // Step 3: Generate meta-features for training set //
    3. Centroid_MFs_train = GenerateMetaFeatures (X_train, Centroids)
    // Step 4: Generate MetaFeatures for Test Set //
    4. Centroid_MFs_test = GenerateMetaFeatures(X_test, Centroids)
End
```

TABLE 4.1: DBMFs Groups Description

Group Name	Description	Number of Dimensions
CentL2	Centroid meta-features generated using Euclidean distance.	C
CentCos	Centroid meta-features generated using cosine similarity.	C
kNNL2	kNN meta-features generated using Euclidean distance.	$C \times k$
kNNCos	kNN meta-features generated using cosine similarity.	$C \times k$
Cent [5]	Concatenation of CentL2 and CentCos.	$2 \times C$
CentkNNCos	Concatenation of Cent and kNNCos.	$C \times (k+2)$
All	Concatenation of Cent, kNNCos, and kNNL2.	$(2 \times C) \times (k+1)$

4.2.4 Embedding property in *Tasneef* and concatenation procedure

4.2.4.1 Pre-trained word embeddings usage

Pre-trained word embeddings allow for quick integration, eliminating the need for resource-intensive training and reducing initial costs. As a result, we chose to work with a set of pretrained fastText version word vectors from fastText library[6] trained on Command Crawl[7] and Wikipedia[8]. This model was trained using CBoW with position-weights, in dimension 300, with character n-grams of length 5, a window of size 5 and 10 negatives. After loading the pretrained word vectors from the fastText library, we obtain the corresponding pretrained word vector for each word in the input text by looking up the vectors associated with each word in the pretrained word vector set. Subsequently, we create a dense vectorization for the entire text by combining these word vectors, typically by averaging them. Algorithm

[5]In all our experiments, we consistently use the two groups 'CentL2' and 'CentCos' as a combined group named 'Cent' due to their minimal feature count and lower computational cost.

[6]https://fasttext.cc/docs/en/support.html

[7]https:https://commoncrawl.org/

[8]https://www.wikipedia.org/

4 describes the document representation using pretrained fast- Text model from fastText library.

4.2.4.2 Concatenation of DBMFs and fasText embeddings

Upon the derivation of DBMFs, we proceed to concatenate them with fastText results in an end-to-end fashion. The feature concatenation process, shown in Figure 4.2, captures document distance relationships via document labels and statistical features, resulting in an initial concatenation between centroid and kNN meta-features (MFs). Subsequently, semantic text attributes are integrated using fastText embeddings. Ultimately, the combined concatenation yields a more concise data representation. Note that DBMFs are considered dense even if they are generated from sparse vectors (such as TF-IDF). Therefore, the final concatenation is being done between two dense vectors.

Algorithm 4 Document Representation Using Pre-trained fastText Model from fasttext Library

```
Input:
    text: document in textual form
Output:
    text_vector: numerical representation of text
Begin
    1. Load FastText library
    2. model = load pre-trained fastText model for Arabic language
    3. Preprocess the text (e.g., tokenization, removing special characters)
            - words = preprocess(text)
    4. Initialize an empty vector for the text representation
            - text_vector = Initialize vector with zeros of appropriate size
    5. Initialize a counter for the number of words
            - word_count = 0
    6. For each word in words:
            a. Get word vector from FastText model
                - word_vector = model.get_word_vector(word)
            b. Add the word vector to the text vector
                - text_vector = text_vector + word_vector
            c. Increment the word counter
                - word_count = word_count + 1
    EndFor
    // Calculate the average vector for the entire text document//
    7. If word_count > 0:
            - text_vector = text_vector / word_count
       Else:
            - Set the text_vector to a zero vector of appropriate size
    EndIf
End
```

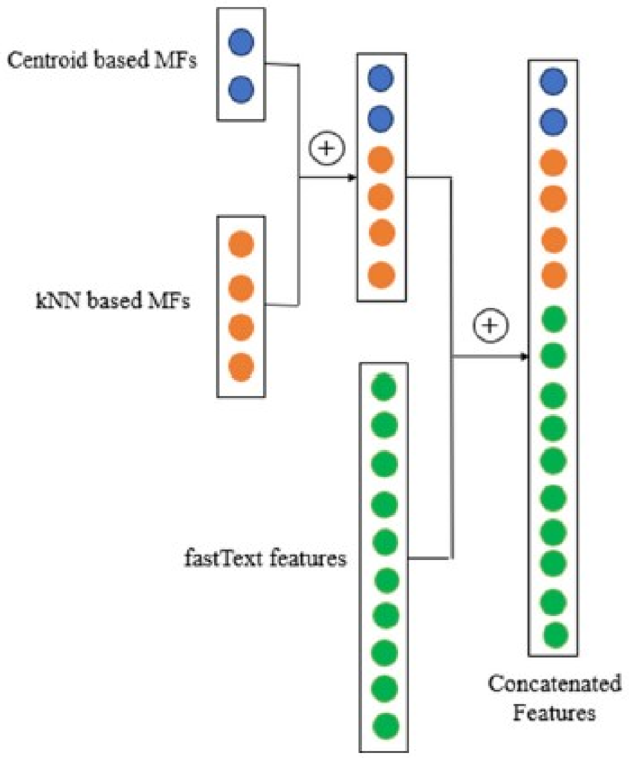


FIGURE 4.2: The feature concatenation procedure in *Tasneef*.

4.3 Experimental setup

4.3.1 Overall architecture

In this Section, we describe the experimental setup involving the evaluation tools and the overall experimental steps. We highlight the main tools used for subsequent comparison: datasets, metrics and *Tasneef* SOTA benchmarks.

4.3.2 Evaluation tools

4.3.2.1 Metrics used

The evaluation of classification methods involves a comprehensive analysis using appropriate performance metrics usually derived from the confusion matrix, such as accuracy, precision, recall, specificity, and F1-score. Additionally, to ensure the reliability and applicability of selected features across diverse datasets or classification models, crossvalidation techniques are implemented. Throughout our First Series of Experiments, we employed two metrics suitable for multi-class classification: Micro-averaged F1 (MicroF1) and Macro-averaged F1 (MacroF1). MicroF1 evaluates classification effectiveness across all decisions, whereas

MacroF1 assesses classification effectiveness for each individual class and computes their average. Both MicroF1 and MacroF1 were utilized in our analysis to provide a comprehensive understanding of classification performance (Sokolova and Lapalme, 2009). In addition, we also used cross-validation methods, dimensionality reduction and runtime evaluations. In the Second Series of Experiments, we employed accuracy and F-Measure metrics, consistent with those used in the chosen benchmarks' works, to ensure a fair and consistent comparison.

4.3.2.2 Datasets used

The proposed method starts with the input of raw text data, provided by online datasets. We rely on seven openly accessible, extensive, single-labelled news documents datasets written in modern standard Arabic: SANAD dataset incorporating Al Arabiya, Al Khaleej, and Akhbarona news portals, in addition to KALIMAT, CNN, Khaleej-2004, and Watan-2004 datasets; all briefly described below.

- **SANAD resource:** The SANAD resource includes three datasets: Al Arabiya, Al Khaleej, and Akhbarona, covering categories such as Culture, Finance, Medicine, Politics, Sports, Religion, and Technology. Al Arabiya omits Culture and Religion. 116 documents were removed from Akhbarona due to lack of textual content or damage. The initial dataset contains 46,900 documents (Einea, Elnagar, and Debsi, 2019).
- **KALIMAT dataset:** KALIMAT dataset, as detailed in (El-Haj and Koulali, 2013) represents a multipurpose Arabic corpus, sourced fromWatan-2004 Arabic corpus. The actual number of articles in KALIMAT is 18,256.13 [9] This corpus is categorized into six distinct sections, covering Culture, Economy, Local News, International News, Religion, and Sports.
- **CNN corpus:** The CNN corpus encompasses 5,070 textual documents. These documents are classified into six categories: Business, Entertainment, Middle East News, Science Technology, Sports, and World News (Saad and Ashour, 2010).
- **Khaleej-2004:** Khaleej-2004 dataset comprises 5,690 documents categorized into four classes: Economy, International News, Local News, and Sports (Sabri, El Beggar, and Kissi, 2024).
- **Watan-2004:** The Watan-2004 Arabic corpus (**a**)contains 20,291 documents, tagged with one of six categories: Culture, Economy, International News, Local News, Religion, and Sports (Abbas, Smaili, and Berkani, 2011).

[9] The authors of the KALIMAT dataset originally reported that it consisted of 20,291 articles, but the actual size we found incorporates 18,256 articles.

Table 4.2 provides an overview of the characteristics of these seven datasets. All were partitioned into training (80%) and testing (20%) sets, for subsequent experimentations.

TABLE 4.2: Characteristics of datasets used.

Dataset	**No. of docs**	**No. of classes**	**Avg. no. of words per doc.**	**Vocabulary size**
Al Arabiya	18,500	5	273	96,247
Al Khaleej	45,500	7	419	239,721
Akhbarona	46,784	7	321	225,901
KALIMAT	18,256	6	978	150,966
CNN Corpus	5,070	6	526	65,752
Khaleej-2004	5,690	4	435	73,368
Watan-2004	20,291	6	528	159,015

4.3.2.3 Benchmarks

For the Second Series of experiments, we carefully selected SOTA methods as described below:

1. **Deep learning models:** We selected ten SOTA deep learning; seven of them are sourced from (Elnagar, Al-Debsi, and Einea, 2020) and include:

   - CNN-based method (Alsaleh and Larabi Marie-Sainte, 2021): CNN.
   - RNN-based method (Goldberg, 2017), including:
     - LSTM-based: CLSTM.
     - Gated-based: GRU, CGRU, BiGRU.
   - HAN-based: HANGRU, HANLSTM.

   Additionally, three **transformer-based language-modeling** methods are sourced from (Azroumahli, Elyounoussi, and Badir, 2023):

   - BERT-based model.
   - MSA model.

- Arabic dialect model.

2. Feature reduction methods: The seven SOTA reduction methods, sourced from (Sabri, El Beggar, and Kissi, 2024) and (Bahassine et al., 2020), include:

   - Feature extraction conventional methods: PCA, LDA.
   - Feature selection methods, including:
     - Information-theoretic-based methods: IG and MI.
     - Statistics-based: chi-square and ImpCHI.
     - Combining statistical and semantic modeling methods: RARF.

4.3.2.4 Hardware used

The experiments were conducted using a Kaggle [10] CPU infrastructure, utilizing a specifically configured Intel® Xeon CPU operating at 2.20 GHz with a total of 30 GB of RAM available. We outline here the overall experimental steps.

4.3.3 Overall experimental steps

In the Initial Experiment, we identify the optimal pair of DBMF groups, based on their average MicroF1 and MacroF1 results. These chosen groups are then combined with fastText to produce the two most effective *Tasneef* variants. Once these variants are determined, we conduct two distinct types of experiments. The First Series of Experiments focuses on comprehensive assessments of accuracy, dimension reduction, and runtime analyses. These analyses are applied to three datasets: Al Arabiya, Al Khaleej, and Akhbarona. The Second Series of experiments compares *Tasneef* variants with SOTA reduction methods and deep learning models using the same datasets and evaluation metrics, including accuracy and F-measure, as employed by those benchmarks. The detailed Experimental Setup is delineated below.

[10] www.kaggle.com

Experimental Setup: Initial Experiment, First Series Experiments, and Second Series Experiments

```
Start
Initial Experiment:
        - Undertake initial hyperparameter tuning.
        - Choose MicroF1 and MacroF1 as metrics.
        - Based on these metrics, select the top two DBMFs from the pool of five,
        listed in Table 4.1.
        - Generate the two most optimal variants of Tasneef by concatenating
        each DBMFs with fastText.
First Series Experiments:
        - Choose three benchmarks: the two Tasneef variants obtained
        above and the top performer among fastText, AraVec, and TF-IDF.
        - For these benchmarks, evaluate:
            •Accuracy:
                ▷ Evaluate Tasneef's accuracy improvement ratio.
            •Dimensionality Reduction:
                ▷ Evaluate Tasneef's reduction ratio.
            •Runtime Analyses:
                ▷ Evaluate total runtime to testing runtime ratio.
                ▷ Evaluate training runtime
                    ✓Evaluate Tasneef's training speedup ratio.
                ▷ Evaluate testing runtime
                    ✓Evaluate Tasneef's testing speedup ratio.
                ▷ Evaluate total runtime
                    ✓Evaluate Tasneef's total speedup ratio.
Second Series Experiments:
        - Choose SOTA reduction methods and deep learning models.
        - Use ad-hoc datasets.
        - Undertake comparisons, considering:
            •Accuracy:
                ▷ Evaluate Tasneef's accuracy improvement ratio.
            •F-Measure:
                ▷ Evaluate Tasneef's F-measure improvement ratio.
Finish
```

4.4 Results and discussions

4.4.1 Initial experiment

The overall results evaluate the quality of texts classification drawn from various datasets. At the initiation of the classification phase, following the acquisition of concatenated vectors, an SVM classifier is utilized. An initial hyperparameter tuning process is conducted to optimize the classifier's performance and ensure robust and accurate classification results.

4.4.1.1 SVM classifier usage

SVM is a powerful classification technique that has been proven effective in Arabic text classification, especially when using the linear SVM as shown in (Abd, Sadiq, and Abbas, 2020). SVM can handle both high-dimensional and low-dimensional feature spaces and is known for its robustness against overfitting. We utilized **LinearSVC** from the scikit-learn [11] library as an SVM classifier with a linear kernel. It is similar to the SVC function with the parameter kernel = 'linear', but implemented in terms of **LIBLINEAR** rather than **LIBSVM**. Note that **LIBLINEAR** is highly efficient on large datasets, as demonstrated in (Fan et al., 2008).

4.4.1.2 Hyperparameters tuning

The kNN-based DBMFs and the linear SVM classifier have hyperparameters that are set prior to training and are not learned from the data. Typically, hyperparameters have the potential to influence the model's performance and its runtime. When tuning hyperparameters, it is common to use grid search and cross-validation. To achieve this, we employed grid search with 5-fold cross-validation to determine the optimal number k of neighbors for generating kNN DBMFs and to find the best value of the SVM regularization parameter C to prevent overfitting. The best k values found experimentally were as follows: k = 60 for the Al Arabiya and Akhbarona datasets, while k = 50 for the Al Khaleej. Due to the variety of methods and datasets, each method paired with a distinct dataset requires a unique optimal parameter C. On the other hand, each dataset typically has only one optimal parameter C. Table 4.3 displays the hyperparameter search grid utilized in *Tasneef*.

[11] `https://scikit-learn.org/stable/`

TABLE 4.3: Hyperparameter search grid.

Hyperparameter	**Value ranges**
k	10, 20, 30,40, 50, 60
C	0.01, 0.1, 1, 10, 100

4.4.1.3 Results of preprocessing

The preprocessing phase follows the insertion of raw text into the pipeline. The raw text corresponds to the datasets outlined in Table 4.2, above. The data preprocessing phase aims to enhance data quality by eliminating noise and unnecessary information, thereby improving the effectiveness of text classification through a reduced vocabulary size. Table 4.4 presents the dataset sizes utilized for subsequent feature processing in each dataset; showing the number of words after preprocessing. The table clearly shows the influence of preprocessing on the reduction in the number of words. As shown, the reduced size ratio varies from 38.50% to 39.92%.

TABLE 4.4: Impact of preprocessing on number of words in dataset.

Dataset	**# of Words Before Preprocessing**	**# of Words After Preprocessing**	**Reduced Size Ratio**
Al Arabiya	5,049,549	3,105,128	38.50%
Al Khaleej	19,073,077	11,614,869	39.12%
Akhbarona	15,033,981	9,032,615	39.92%

4.4.2 Selection of the best DBMFs groups

4.4.2.1 DBMFs baslines choice

A baseline is selected to define the minimum level of performance that any new model must surpass to be considered beneficial. To evaluate the performance of the resulting concatenated DBMFs (kNN-based DBMFs + Centroid-based DBMFs), each DBMFs group is initially selected independently as a candidate. A comprehensive description of all DBMFs groups is provided in Table 4.1. We use MicroF1 and MacroF1, described above, as metrics for ranking these baselines.

a) **Statistical significance**

We performed five multiple random test-train comparisons.The reported results of all our subsequent experiments (in Table 4.5 and 4.6) are based on the average performance MicroF1 and MacroF1 metrics applied to the test set. This approach helps to assess model stability and provides a more reliable estimate of the model's generalization performance. Furthermore, we applied a paired t-test to assess the significance of the results obtained in our experiments using a 95% confidence level. To address the issue of conducting multiple tests simultaneously, we apply the Bonferroni correction, detailed in the Appendix of the thesis. This correction is employed to control the familywise error rate and maintain the overall significance level when making multiple comparisons (Haynes, 2013).

b) **Ranking procedure**

We require a ranking procedure to select the two most optimal DBMFs to be concatenated with fastText, thereby creating two *Tasneef* variants. The ranking results, reported in Table 4.5, show DBMFs groups performance based on MicroF1 and MacroF1 metrics using paired t-test with Bonferroni correction. The ranking procedure, used in Table 4.5 is based on the following steps:

1. Evaluate all DBMFs using MicroF1 and MacroF1.
2. Choose the best DBMF group as a reference; here, the DBMF 'All' is taken as a reference because it exhibits the best results across all datasets. Put this reference DBMF group at the top of the table.
3. For each dataset, rank the remaining DBMFs with respect to this reference, from closest to the farthest.
   a. Use paired t-tests with Bonferroni correction (See Appendix of the thesis for more details).
      i. Indicate statistical ties with bidirectional arrows.
      ii. Indicate statistically significant losses with downward arrows.
   b. Remove from competition all values with losses because the results must include statistically significant ties only.
4. For each column, highlight the three best values, at most.
5. Keep only the best DBMF, in addition to the reference.

TABLE 4.5: Meta-features groups performance based on MicroF1 and MacroF1 metrics using paired t-Test with Bonferroni correction.

	Al Arabiya		Al Khaleej		Akhbarona	
	MicroF1	MacroF1	MicroF1	MacroF1	MicroF1	MacroF1
All Ref. DBMF *	**97.610**	**97.609**	**96.920**	**96.918**	**94.243**	**94.235**
CentkNNCos	**97.481** ⇔	**97.479** ⇔	**96.837** ⇔	**96.834** ⇔	**94.192** ⇔	**94.183**⇔
Cent	**97.162** ⇔	**97.159** ⇔	95.490 ⇓	95.483 ⇓	91.980 ⇓	91.968 ⇓
kNNL2	97.059 ⇓	97.056 ⇓	96.098 ⇓	96.093 ⇓	93.602 ⇓	93.589 ⇓
kNNCos	97.059 ⇓	97.057 ⇓	96.230 ⇓	96.224 ⇓	93.649 ⇓	93.636 ⇓

TABLE 4.6: *Tasneef* comparison with other benchmarks using paired t-test with Bonferroni correction.

		Al Arabiya		Al Khaleej		Akhbarona	
		MicroF1	MacroF1	MicroF1	MacroF1	MicroF1	MacroF1
Ref. *	TF-IDF	**98.243**	**98.242**	**97.841**	**97.840**	**95.051**	**95.046**
Tasneef variants	*Tasneef_var2*	**98.054** ⇔	**98.053** ⇔	**97.714** ⇔	**97.713**⇔	**94.910**⇔	**94.904**⇔
	Tasneef_var1	**97.935**⇔	**97.934**⇔	**97.705**⇔	**97.704**⇔	**94.972** ⇔	**94.965**⇔
Other benchmarks	fastText	**97.848**⇔	**97.846**⇔	96.934⇓	96.933⇓	93.760⇓	93.755⇓
	AraVec	97.310⇓	97.310⇓	96.140⇓	96.141⇓	92.606⇓	92.609⇓

* In both Table 4.5 and Table 4.6, note that the first line is taken as a reference; consequently, no arrows are used. For more details regarding the threshold calculation between ties and losses, refer to the Appendix 4.5.

4.4.2.2 DBMFs ranking results

a) **Best DBMFs groups**

According to the results outlined in Table 4.5, it is clear that 'All' DBMFs and 'CentkNNCos' DBMFs emerge as the best two groups across all datasets. Furthermore, Al Arabiya dataset demonstrates the highest performance, closely followed by the Al Khaleej dataset, with the Akhbarona dataset exhibiting slightly lower performance. It is also worth mentioning that MicroF1 and MacroF1 values are remarkably similar: a characteristic attributed to the relative balance of the datasets used.

b) ***Tasneef* variants obtainment**
Now, all we need is the concatenation of these two best DBMFs with fastText. We will consequently use the two variants of*Tasnef*, i.e., 'All+fastText', referred to as *Tasnef*_var1, and 'CentkNNCos+fastText', and referred to as *Tasnef*_var2. Note that the symbol '+' here denotes concatenation. We are now ready for the First Series of Experiments.

4.4.3 First series of experiments: baselines performance

Upon selecting the two optimal *Tasneef* variants, we conduct a comparative analysis with TF-IDF, AraVec, and fastText, utilizing the same ranking procedure employed for the derivation of the *Tasneef* variants, as detailed in Table 4.6. Note that for a fair comparison, we applied the same preprocessing procedure and used SVM as the default classifier for all methods.

4.4.3.1 MicroF1 and MacroF1 results

a) **Retained benchmarks**
Table 4.6 presents the performances of *Tasneef* variants, TFIDF, AraVec, and fastText. We observe that the two *Tasneef* variants achieve the best results, similar to TF-IDF, across all datasets based on statistical significance comparisons. Excluding TF-IDF, it is evident that *Tasneef_var2* surpasses the second-best benchmark method (fastText) by 0.21%, 0.78%, 1.12% on the Al Arabiya, Al Khaleej, and Akhbarona datasets, respectively. TF-IDF will be retained as the sole competitor of *Tasneef* variants for further comparisons.

b) **Comparison at dataset level**
At the dataset level, consistent with the previously mentioned findings, Al Arabiya dataset displays the highest performance, closely followed by Al Khaleej dataset. However, the results for Akhbarona dataset indicate a somewhat lower level of performance. Another noteworthy observation is that fastText, on its own, outperforms AraVec across all datasets. This can be attributed to fastText's capability to work with subword information by considering character n-grams. This feature renders fastText highly effective, particularly in handling morphologically rich languages like Arabic.

c) **TF-IDF vs. *Tasneef***

Our two variants closely approximate TF-IDF, within a maximum loss of 0.189%, compared to TF-IDF for both MicroF1 and MacroF1 metrics on the Al Arabiya dataset, 0.127% on the Al Khaleej dataset, and 0.08% on the Akhbarona dataset. However, as Table 6 shows, the paired t-test with Bonferroni correction indicates that this loss is not statistically significant, meaning the performance difference between our two variants and TF-IDF could be due to random variation rather than a true effect.

TF-IDF combined with a finely tuned SVM serves as a robust vectorization benchmark for our datasets, making it challenging to surpass using *Tasneef*, at least for our specific datasets. However, *Tasneef* offers substantial improvements in terms of memory efficiency and runtime performance.

4.4.3.2 Dimentionality reduction in *Tasneef*

Table 4.7 reports the dimensionality reduction results for the two *Tasneef* variants and TF-IDF, specifying the number of dimensions (labeled as 'Dim.' in the table) and reduction ratios ('Reduc.') compared to TF-IDF across all datasets. The dimension reduction ratio is calculated between the original dimension given by TF-IDF (for instance 96,247 for the Al Arabiya dataset) and the dimension generated by any *Tasneef* variant (such as 910 for *Tasneef_var1*), yielding a reduction ratio of 106(=96,247/910). It is shown that *Tasneef_var2* achieves substantial dimensionality reduction, with dimension reduction ratios of 158x, 361x and 308x, compared to TF-IDF, across Al Arabiya, Al Khaleej, Akhbarona, respectively. Similarly, the other variant of *Tasneef*, (i.e., *Tasneef_var1*),reduces TF-IDF dimension by factors of 106x, 236x, and 196x, with respect to the same datasets. As an overall result, the dimension reduction of *Tasneef* is a two-order magnitude, and ranges from 158x to 361x, across all three datasets, compared to TF-IDF. As shown in Figure 4.3, *Tasneef* achieves a substantial two-order of magnitude dimension reduction compared to TF-IDF across all three datasets, with reduction ratios ranging from 158x to 361x.

TABLE 4.7: *Tasneef* dimensionality reduction as compared to TF-IDF.

		Al Arabiya		Al Khaleej		Akhbarona	
		Dim.	Reduc.	Dim.	Reduc.	Dim.	Reduc.
Tasneef variants	*Tasneef_var1*	910	106x	1014	236x	1154	196x
	Tasneef_var2	610	**158x**	664	**361x**	734	**308x**
Benchmark	TF-IDF	96,247	-	239,721	-	225,901	-

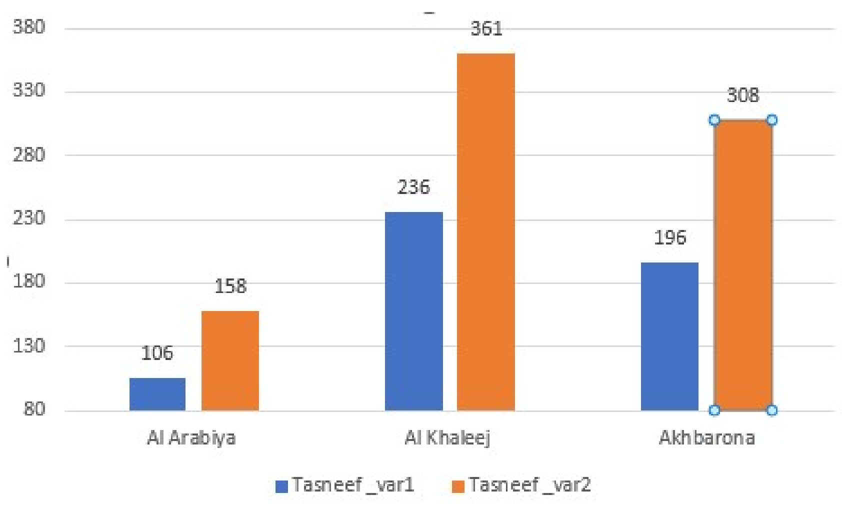


FIGURE 4.3: *Tasneef* dimension reduction ratios w.r.t. TF-IDF

4.4.3.3 Runtime analyses

Table 4.8 describes the runtime analyses. The values mentioned in the three columns under each dataset represents respectively,the total time (Total), the SVM learning time(SVM Lean.), both in minutes, and SVM testing time(SVM Test.), in milliseconds (ms). The total time includes preprocessing, DBMFs generation, concatenation with word embeddings, and classification runtimes, including hyperparameter tuning with cross-validation, training, and testing.

TABLE 4.8: *Tasneef* runtime results.

	Al Arabiya			Al Khaleej			Akhbarona		
	Total (min)	SVM Learn. (min)	SVM test. (ms)	Total (min)	SVM Learn. (min)	SVM test. (ms)	Total (min)	SVM Learn. (min)	SVM test. (ms)
Tasneef_var1	6.87	3.13	7	29.75	17.10	18	36.28	24.35	13
Tasneef_var2	**3.64**	**1.3**	**6**	**20.67**	**7.65**	**11**	**24.56**	**11.98**	**11**
TF-IDF	4.50	3.76	837	28.44	25.62	5765	30.17	27.85	5361

a) **Total runtime**

As indicated in Table 4.6, the analysis revealed that *Tasneef_var1*, *Tasneef_var2* and TF-IDF, yielded the most favorable accuracy results. However, when examining runtime, a clear preference emerged for one *Tasneef* variant, namely *Tasneef_var2*, as the fastest among the three benchmarks.

b) **Speedup ratio of *Tasneef* variants**

For a specified dataset (like Al Arabiya, for instance), the speedup ratio of any runtime (such as total runtime, testing runtime, or learning runtime) of the *Tasneef* variant, relative to a specified method (for instance, TF-IDF), is computed as the ratio between the runtime of the specified method (in this case, TF-IDF) and the corresponding runtime of the *Tasneef* variant. For example, for Al Arabiya dataset, the total runtime, in minutes, of TF-IDF is 4.50 and the corresponding total runtime of the *Tasneef_var2* is 3.64. In this case, the speedup ratio of the total runtime of *Tasneef_var2* would be 4.50/3.64 (=1.24). This means that the total runtime of *Tasneef_var2* is shorter than TF-IDF by 24%. Moreover, this *Tasneef* variant demonstrates a significantly enhanced speed in total runtime compared to the alternative, resulting in a speedup ratio of 1.38 (a 38% enhancement) for the Al Khaleej dataset and 1.23 (a 23% improvement) for the Akhbarona dataset, when compared to the TF-IDF method. These findings emphasize the efficiency of *Tasneef_var2* in terms of total runtime across all datasets. Furthermore, the alternative *Tasneef_var1*, which exhibits lower performance, is enhanced by *Tasneef_var2* with speedup ratios of 1.89, 1.44, and 1.48 for the same datasets, listed in the same order as previously. Figure 4.4 illustrates the speedup ratio in total runtime enhancement of the *Tasneef_var2* in comparison to TF-IDF and *Tasneef_var1*, across all datasets. *Tasneef_var2* improves TF-IDF total runtime by 24%, 38%, 23%, on Al Arabiya, Al Khaleej, Akhbarona, respectively. Furthermore, *Tasneef_var2* enhances *Tasneef_var1* by 89%, 44%, and 48% for the same datasets, listed in the same order as above.

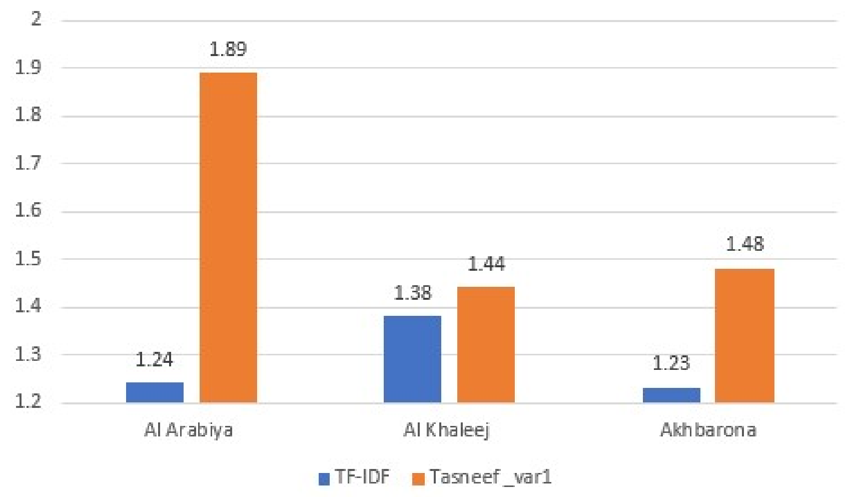


FIGURE 4.4: Speedup ratio of total runtime of the fastest *Tasneef* variant (i.e., *Tasneef_var2*)

c) **Training runtime**

- **For TF-IDF:** The training runtime spent by the SVM classifier, including hyper-parameter tuning and actual learning, consumes 84%, 90%, and 92%, of the total time when using TF-IDF, for Al Arabiya, Al Khaleej, and Akhbarona datasets, respectively.
- **For *Tasneef_var1*:** The training runtime represents 46%, 57%, and 67% of the total time, for Al Arabiya, Al Khaleej, and Akhbarona datasets, respectively.
- **For *Tasneef_var2*:** The training runtime is 36%, 37%, and 49% of the total time, for Al Arabiya, Al Khaleej, and Akhbarona datasets, respectively. Once again, *Tasneef_ var2* produces the best ratios.

From Figure 4.5, it is evident that both variants of *Tasneef* give the least ratio of training runtime to total runtime, meaning that they are the fastest in learning.

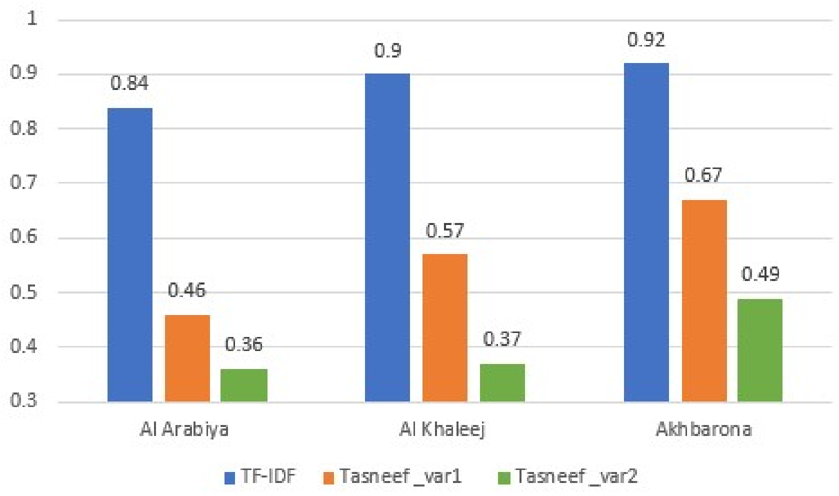


FIGURE 4.5: Ratio of training runtime to total runtime of all chosen baselines.

d) **Runtime of testing phase**

During the testing phase, the runtime is measured in milliseconds rather than minutes. From Table 4.8, we can easily note that the runtime required to test the SVM model using *Tasneef_var1* improves the runtime performance of TF-IDF by factors of 120x, 320x, and 412x, for Al Arabiya, Al Khaleej, and Akhbarona datasets, respectively, compared to TF-IDF. Similarly, *Tasneef_var2* improves the runtime performance of TF-IDF by factors of 140x, 524x, and 487x times faster, on the same datasets above, respectively. These results suggest that *Tasneef* improves the runtime of the testing phase of TF-IDF by a two orders of magnitude, ranging from 120x to 524x. Figure 6 shows *Tasneef* speedup ratio of testing phase runtime of both *Tasneef* variants with respect to TFIDF. *Tasneef* improves TF-IDF's testing phase runtime by two orders of magnitude, ranging from 120x to 524x. These promising results are due to dimension reduction.

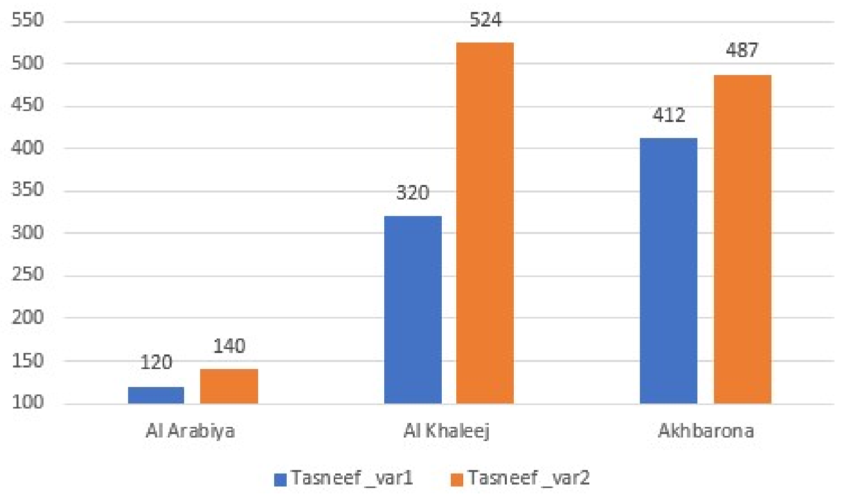


FIGURE 4.6: *Tasneef* speedup ratio of testing phase runtime w.r.t. TF-IDF.

4.4.4 Second series of experiments: comparison with SOTA methods

Utilizing the Experimental Setup outlined in Subsection 4.4.1 above, we proceed to compare *Tasneef_var2*, identified as the optimal variant of *Tasneef*, with SOTA reduction methods and deep learning models. Our evaluation includes two standard metrics directly taken from existing literature: accuracy and F-measure. The datasets outlined in Table 4.2 serve as the basis for this comparison. It is important to mention that, for all subsequent experiments, we have selected only the top four performing deep learning models from the initial pool of nine. This selection process was carried out independently for each of the three datasets (Al Arabiya, Al Khaleej, and Akhbarona) proposed in the study reported in (Elnagar, Al-Debsi, and Einea, 2020) and shown on Table 4.9 As explained in Section 4.3.2.3, above, the overall number of all benchmarks is seventeen: ten from deep learning methods and seven from dimension reduction methods.

4.4.4.1 *Tasneef_var2* accuracy improvement ratio (AIR)

Table 4.9 shows the accuracy results of all benchmarks. We assess the improvement yielded by *Tasneef_var2* using the accuracy improvement ratio (AIR), for each dataset. The AIR is defined as the percentage by which *Tasneef_var2* enhances the accuracy of a specific method, for a given dataset. The AIR given by the following formula:

$$\text{AIR} = \left(\frac{\text{Tasneef\_Accuracy}}{\text{Competitor\_Accuracy}} - 1\right) \times 100 \tag{4.4}$$

For instance, considering the Akhbarona dataset and the CLSTM method from Table 4.9, *Tasneef_var2* achieves an accuracy of 95.43% while CLSTM attains 92.66%. This yields a *Tasneef_var2* AIR of 3.0% $[(= 95.43/92.66) - 1) \times 100]$, indicating a 3.0% enhancement in CLSTM accuracy. Alternatively, by considering the CLSTM accuracy, we can determine the *Tasneef_var2* accuracy as follows: $(92.66 + (92.66 \times 3.0\%)) = 95.43$. As a whole, for *Tasneef_var2*, the AIR ranges from a minimum of 0.3% with respect to (RARF+SVM) on the Al Khaleej-2004 dataset, to a maximum of 39.6% with respect to LDA+SVM on the Watan-2004 dataset. Furthermore, on the KALIMAT dataset, our method demonstrates significant AIR, achieving a notable 14.9% enhancement over the best-pretrained BERT model (BERT-based MSA model).

Note that the *Tasneef's* accuracy values, found in this Second Series of Experiments are slightly different from those of the First Series of Experiments, reported above. The accuracies in percentage are now 98.43 (instead of 98.05, above), 97.49 (instead of 97.71), and 95.43 (instead of 94.91) for Al Arabiya, Al Khaleej, and Akhbarona, respectively. The key difference lies in our approach to training *Tasneef_var2*. Here, we utilized the same training and testing sets as the deep learning models referenced in (Elnagar, Al-Debsi, and Einea, 2020), ensuring a fair comparison of results between our model and theirs. In the First Series Experiments, the reported results of all our experiments are based on the average MicroF1 performance of multiple random test-train splits (five splits). The results are therefore slightly different. The overall results highlight *Tasneef_var2*'s AIR across all six datasets, as shown on Table 4.9 and highlighted on Figure 4.7 and Figure 4.9.

TABLE 4.9: *Tasneef* accuracy improvement ratio w.r.t. deep learning models and reduction methods.

Dataset	**Ref.**	**Method**	**Acronym meaning**	**Accuracy**	***Tasneef* Acc. Improvement Ratio (%)**
Al Arabiya	(Elnagar, Al-Debsi, and Einea, 2020)	BiGRU	bidirectional gated recurrent units	97.41	1.0
		CGRU	convolutional gated recurrent units	97.19	1.3
		CLSTM	contextual long short-term memory	96.97	1.5
		GRU	gated recurrent units	96.76	1.7
		Tasneef_var2	**-**	**98.43**	**-**

Continued on next page

Al Khaleej	(Elnagar, Al-Debsi, and Einea, 2020)	CGRU	convolutional gated recurrent units	96.86	0.7
		HANGRU	hierarchical attention network gated recurrent units	96.66	0.9
		CLSTM	contextual long short-term memory	96.59	0.9
		HANLSTM	hierarchical attention network long short-term memory	96.55	1.0
		Tasneef_var2	-	**97.49**	-
Akhbarona	(Elnagar, Al-Debsi, and Einea, 2020)	CGRU	convolutional gated recurrent units	94.00	1.5
		HANGRU	hierarchical attention network gated recurrent units	92.95	3.0
		CNN	convolutional neural network	92.72	3.0
		CLSTM	contextual long short-term memory	92.66	3.0
		Tasneef_var2	-	**95.43**	-

Continued on next page

Khaleej-2004	(Sabri, El Beggar, and Kissi, 2024)	PCA+SVM	principal component analysis + support vector machine	93.66	1.5
		LDA+SVM	linear discriminant analysis + support vector machine	74.52	27.5
		Chi-square + SVM	chi-square + support vector machine	94.20	0.9
		MI+SVM	mutual information + support vector machine	93.94	1.2
		RARF+SVM	removal of Arabic redundant features + support vector machine	94.75	0.3
		Tasneef_var2	**-**	**95.07**	**-**

Continued on next page

Watan-2004	(Sabri, El Beggar, and Kissi, 2024)	PCA+SVM	principal component analysis + support vector machine	92.15	2.9
		LDA+SVM	linear discriminant analysis + support vector machine	67.90	39.6
		Chi-square + SVM	chi-square + support vector machine	92.89	2.1
		MI+SVM	mutual information + support vector machine	93.94	1.2
		RARF+SVM	removal of Arabic redundant features + support vector machine	94.75	0.3
		Tasneef_var2	**-**	**95.07**	**-**

Continued on next page

KALIMAT	(Azroumahli, Elyounoussi, and Badir, 2023)	BERT-based Model	bidirectional encoder representations from transformers-based Model	82.60	14.9
		MSA Model	modern standard Arabic model	85.40	11.1
		Arabic Dialect Model	-	77.20	22.9
		Tasneef_var2	**-**	**94.90**	**-**

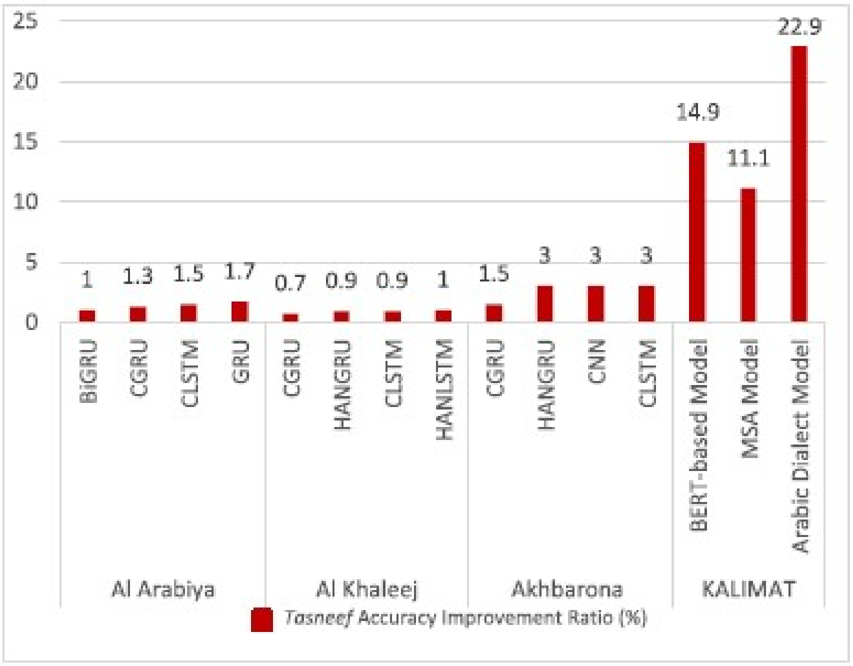


FIGURE 4.7: *Tasneef* accuracy improvement ratio (in %) w.r.t. deep learning models on Al Arabiya, Al Khaleej, Akhbarona and KALIMAT datasets.

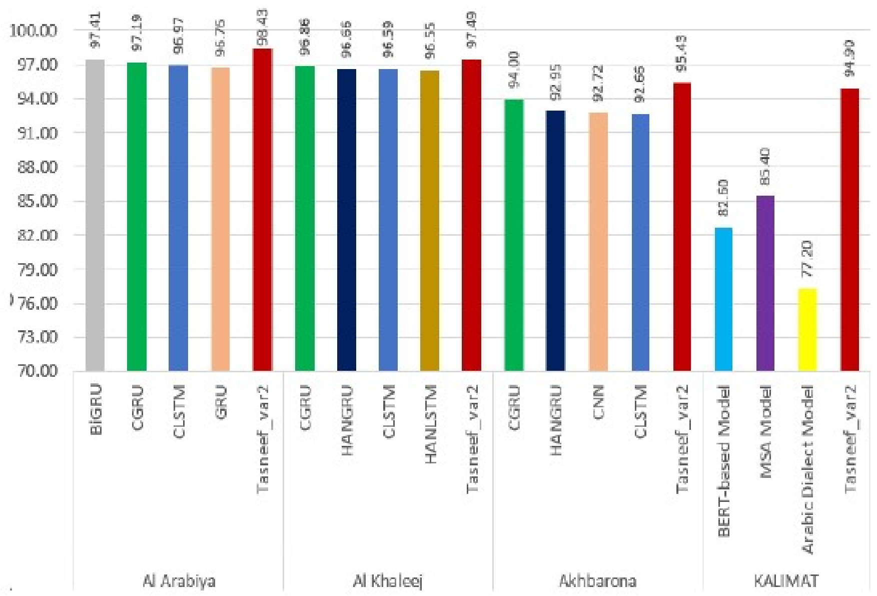


FIGURE 4.8: The accuracy of *Tasneef* compared to deep learning benchmarks.

Figure 4.7 shows the *Tasneef* AIR (in %) with respect to all deep learning models, on Al Arabiya, Al Khaleej, Akhbarona and KALIMAT datasets. *Tasneef* outperforms deep learning models' accuracy by a ratio ranging between 0.7% (w.r.t. CGRU) and 22.9.0% (w.r.t. Arabic dialect model). Figure 4.8 shows the accuracy of *Tasneef* compared to deep learning benchmarks.

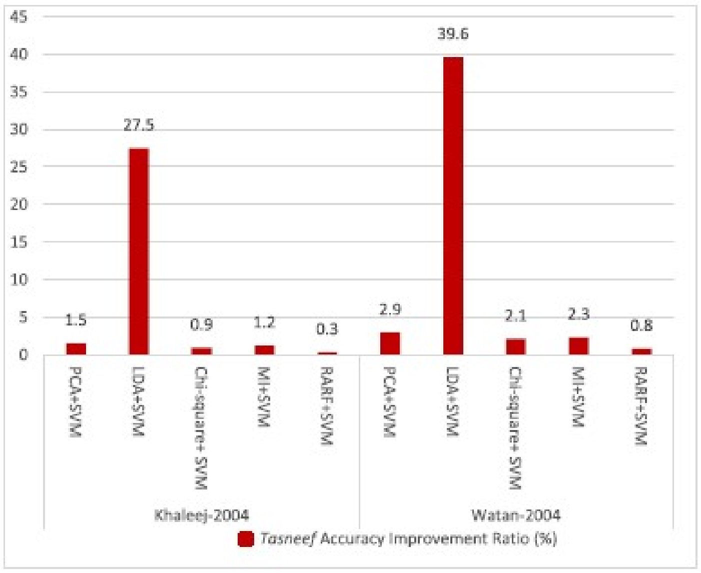


FIGURE 4.9: *Tasneef* accuracy improvement ratio (in %) w.r.t. dimentionality reduction methods on Khaleej-2004 and Watan-2004 datasets.

Figure 4.9 shows the *Tasneef* AIR across Khaleej-2004, Watan-2004 datasets with respect to all chosen reduction methods. The results on Khaleej-2004 and Watan-2004 are based on (Sabri, El

Beggar, and Kissi, 2024). *Tasneef* improves benchmarks' accuracy with a ratio ranging from 0.30% (w.r.t. RARF+SVM) to 39.6% (w.r.t. LDA+SVM).

4.4.4.2 *Tasneef_var2* F-measure improvement

Table 4.10 shows the F-Measure results of the best variant of *Tasneef* and the SOTA reduction methods. The comparisons are based on CNN dataset and the results provided in (Bahassine et al., 2020). We note that *Tasneef* improves the F-Measure of SOTA reduction methods by factor ranging from 4.6% (w.r.t.ImpCHI+SVM) to 26.8% (w.r.t. MI+SVM).

TABLE 4.10: *Tasneef* F-measure w.r.t. other reduction methods.

Dataset	**Ref.**	**Method**	**Acronym meaning**	**F-measure**	**Tasneef F-measure Improvement Ratio (%)**
CNN	(Bahassine et al., 2020)	Chi-square + SVM	chi-square + support vector machine	88.10	7.5
		IG+SVM	information gain + support vector machine	88.60	6.9
		ImpCHI + SVM	improved chi-square + support vector machine	90.50	4.6
		MI+SVM	mutual information + support vector machine	74.70	26.8
		Tasneef_var2	-	**94.69**	-

Figure 4.10 shows the improvements of *Tasneef* in F-Measure (in %) with respect to all benchmarks. *Tasneef* enhances benchmarks' F-measure by ratios ranging from 4.6% (w.r.t. ImpCHI+SVM) to 26.8% (w.r.t. MI+SVM).

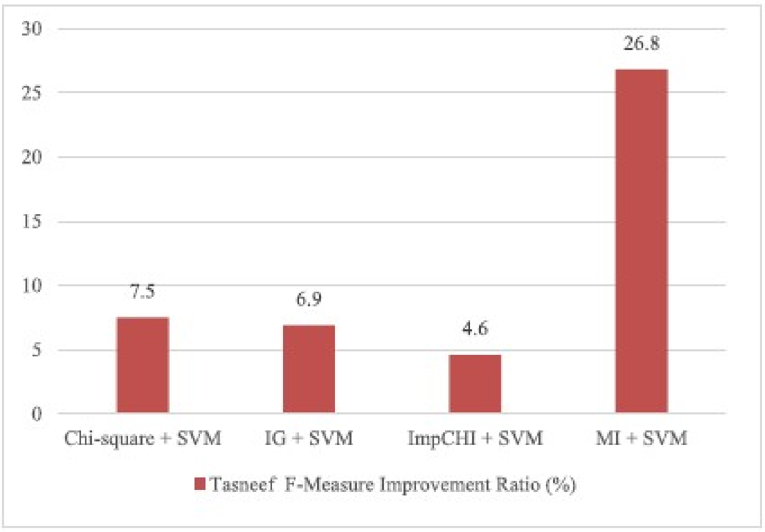


FIGURE 4.10: *Tasneef* F-measure improvement ratio (in %) compared to SOTA reduction methods, on the CNN dataset.

4.4.5 Summary of *Tasneef* main improvements

We now summarize the main results. Table 4.4.5 shows the main contributions of *Tasneef*. The minimum and maximum gains are reported. We also highlight the losses of the method, solely with respect to TF-IDF.

Algorithm 1 *Tasneef* method for Arabic Text Classification

```
Input: Raw document $D$
Output: Classification of $D$ into one class
Begin
    // Raw data //
    1. Load raw data $D$

    // Preprocessing //
    2. Clean document $D$
        a. Remove stop-words
        b. Remove special symbols, numerals, etc.
    3. Apply tokenization on $D$: call PyArabic tokenizer
    4. Apply stemming algorithm on $D$: call ISRI stemmer from the NLTK library
    5. Obtain cleaned document $D_c$

    // Statistical Property //
    6. Vectorization and reduction of $D_c$
        a. TF-IDF: Transform $D_c$ into sparse feature vector ($V_F$)
        b. Dimension reduction of $V_F$: use DBMFs
            i. Local reduction: use kNNL2 and kNNCos
            ii. Global reduction: use CentL2 and CentCos
        c. Obtain reduced dense feature vector ($V_{F_{reduced}}$)

    // Embeddings Property //
    7. Use pre-trained fastText embedding
    8. Generate fastText embeddings and obtain feature vector ($V_{F_{fastText}}$)

    // Concatenation & Results //
    9. Concatenate results obtained in steps 6c and 8
        i. Obtain ($V_{F_{concatenated}}$)
        ii. Input ($V_{F_{concatenated}}$) into SVM
    10. Classify $D$
End
```

Metric	***Tasneef* Losses vs. Improvements**
MicroF1 / MacroF1	Minimum loss for *Tasneef*: *Tasneef_var2* lags TF-IDF by 0.079%, on Akhbarona dataset.
	Maximum loss for *Tasneef* : *Tasneef_var2* lags TF-IDF by 0.189%, on Al Arabiya dataset.
	Minimum gain with respect to fastText: *Tasneef_var2* surpasses it by 0.21%, 0.78%, 1.12% on the Al Arabiya, Al Khaleej, and Akhbarona datasets, respectively.
	Maximum gain with respect to fastText: *Tasneef_var2* surpasses it by 0.21%, 1.08%, 1.15% on the same datasets, respectively.
	Minimum gain with respect to AraVec: *Tasneef_var2* surpasses it by 0.74%, 1.57%, 2.36% on same datasets, respectively.
	Maximum gain with respect to AraVec: *Tasneef_var2* surpasses it by 0.75%, 1.58%, 2.37% on the same datasets, respectively.
Runtime of Testing Phase	Minimum: *Tasneef_var1* improves TF-IDF by 120x, on Al Arabiya dataset.
	Maximum: *Tasneef_var2* improves TF-IDF by 524x, on Al Khaleej dataset.
Training Runtime	Minimum: *Tasneef_var2* improves TF-IDF by 23%, on Akhbarona dataset.
	Maximum: *Tasneef_var2* improves TF-IDF by 38%, on Al Khaleej dataset.
Training Runtime to Total Time Ratio	Minimum: *Tasneef_var2* consumes 36% of Total Time for training, on Al Arabiya dataset; compared with 84% for TF-IDF.
	Maximum:*Tasneef_var1* consumes 67 of Total Time for training, on Akhbarona dataset; compared with 92% for TF-IDF.

Continued on next page

Dimension Reduction (Memory conciseness)	Minimum: *Tasneef_var1* improves TF-IDF by 106x, on Al Arabiya dataset.
	Maximum: *Tasneef_var2* improves TF-IDF by 361x, on Al Khaleej dataset.
Accuracy	Minimum: *Tasneef_var2* improves RARF+SVM method by 0.3%, on Al Khaleej-2004 dataset.
	Maximum: *Tasneef_var2* improves LDA+SVM method by 39.6%, on Watan-2004 dataset.
F-measure	Minimum: *Tasneef_var2* improves ImpCHI+SVM method by 4.6%, on CNN dataset.
	Maximum: *Tasneef_var2* improves MI+SVM method by 26.8%, on CNN dataset.

4.5 Conclusion

In this chapter, we presented a simple yet effective hybrid approach, *Tasneef*, designed to overcome the computational challenges associated with Arabic Text Classification (ATC). By merging DBMFs with fastText embeddings, this approach significantly enhances both memory efficiency and runtime performance without compromising classification accuracy. *Tasneef's* innovative design incorporates both local and global information through kNN and centroid-based DBMFs, respectively. This combination shows a comprehensive representation of text, capturing statistical relationships while integrating semantic properties via word embeddings. The empirical evaluation across seven prominent Arabic datasets highlights *Tasneef's* ability to reduce memory usage by up to 361 times and runtime by up to 524 times compared to traditional methods. Moreover, *Tasneef* maintains comparable MicroF1 and MacroF1 values, achieving an accuracy improvement ranging from 0.3% to 39.6% and an F-Measure enhancement from 4.6% to 26.8% across various datasets. These results highlights *Tasneef's* potential as a robust and efficient solution for ATC, capable of handling large-scale data with reduced computational costs. Furthermore, *Tasneef's* superior performance against ten SOTA deep learning models and seven dimensionality reduction methods demonstrates its effectiveness and efficiency in real-world applications.

Conclusion

Text classification, the task of assigning a document to a predefined class, is fundamental in natural language processing. While machine learning offers effective frameworks for this task, deep learning achieves high accuracy but often incurs significantly higher computational costs. Although numerous algorithms exist for English text classification, research on Arabic remains limited due to its linguistic complexity and the substantial computational resources required. The research objectives aimed to bridge the "digital gap" in ATC by enhancing computational efficiency through dimensionality reduction methods, striving to improve or at least maintain classification effectiveness.

In this dissertation, we reviewed existing ATC approaches, including preprocessing methods, vectorization strategies, dimensionality reduction techniques, and both classical machine learning and deep learning models. This analysis of strengths and limitations served as the foundation for proposing three primary methods: The first proposed method was PCA, which serves as a feature extraction technique for reducing the feature space size. To assess the impact of this method on the classification process, we evaluated it using five popular classifiers: logistic regression, k-nearest neighbors, decision trees, random forest, and support vector machines. The results demonstrate that the proposed method significantly improves classification accuracy for most classifiers, while also reducing execution time. The second proposed approach sought to address two issues associated with the TF-IDF text representation: high dimensionality and sparsity. These challenges lead to large storage requirements, high computational costs, and the potential for overfitting. To mitigate these issues, four distance-based meta-features (CosKNN, L2KNN, CosCent, and L2Cent) were derived from the TF-IDF representations and used as a dimensionality reduction method for ATC. The impact of these meta-features on the performance of various classification algorithms, including K-Nearest Neighbors, Logistic Regression, Support Vector Machines, and Random Forest, was evaluated. The results demonstrate that the proposed method enhanced classification accuracy in 50% of the cases, while also achieving a substantial reduction in training time—up to 3244 times faster—across all classifiers. Additionally, it outperformed PCA in terms of both accuracy and efficiency. The last proposed method *Tasneef*, a novel hybrid approach developed to address the computational challenges in ATC. By combining DBMFs with fastText embeddings, it improves memory efficiency and runtime performance without sacrificing classification accuracy. *Tasneef* integrates both local and global information through kNN and centroid-based DBMFs, capturing statistical relationships and semantic properties via word embeddings. Empirical evaluations on seven Arabic datasets show that *Tasneef* reduces memory usage by up to 361 times and runtime by up to 524 times compared to traditional methods. Additionally, it maintains comparable MicroF1 and MacroF1 scores, achieving accuracy improvements of up to 39.6% and F-Measure gains from 4.6% to 26.8%. *Tasneef* outperforms ten SOTA deep learning models and seven dimensionality reduction methods, demonstrating its potential as an efficient and robust solution for ATC in large-scale data processing.

In future work, the following points should be considered:

- Using Incremental PCA, which is well-suited for streaming data, or Randomized PCA, which offers faster approximations, could enhance both processing efficiency and scalability, especially for large-scale ATC applications.

- Investigation of the effectiveness of DBMFs on short texts such as social media posts, search queries, and text messages, which are often challenging in text classification due to their limited context and lack of rich semantic information compared to longer texts.
- Introduction of two additional steps after DBMFs generation:the first step involves sparsification to reduce the density of DBMFs representation, minimizing training costs and noise. The second step is selective sampling, which removes certain documents from the matrix by carefully selecting the most relevant ones for the learning phase.
- Exploration of the generalizability and scalability of *Tasneef*, especially for other languages and low-resource settings, across datasets with diverse class distributions, to evaluate its adaptability in different contexts.

Appendix A

Paired t-tests with Bonferroni correction, applied in Chapter 4

In this appendix, we present a detailed explanation of the results from the paired t-tests with Bonferroni correction, as reported in Tables 4.5 and 4.6 of Chapter 4. The method is employed to control the familywise error rate when conducting multiple statistical comparisons, ensuring that the probability of committing at least one type I error (false positive) remains within the desired significance level. The implementation of the Bonferroni correction is predicated on the null hypothesis, which posits no significant difference between the compared groups.

- **Null hypothesis H0:** For each of the m statistical comparisons (tests), assume there is no significant difference between performance scores. This implies, in our case, that for each test, the mean performance score (MicroF1 or MacroF1) of the 'All' group is equal to that of the other group.

- **p-values evaluation:** The p-value is the probability that the observed differences between pairs happened by chance.

1. In the trivial case of only one test:
   If the p-value is less than the original significance level α (commonly $\alpha = 0.05$), the null hypothesis is rejected and we conclude that there is a statistically significant difference between the 'All' group and the other group.

2. In the case of multiple tests:
   we use the correction by adjusting either the significance level α or the p-value.

   a. Adjust α:
      Divide the significance level by the number of tests $\alpha_{\text{adj}} = \alpha/m$. If the p-value is less than α_{adj}, then we reject the null hypothesis (loss); otherwise, we accept it (tie).

   b. Adjust p-value:
      Multiply the p-value by m. If the p-value is less than α, we reject the null hypothesis (loss); otherwise, we accept it (tie). We used this method in all our calculations.

To compare five groups (All, CentKNNCos, Cent, kNNCos, and kNNL2), we performed ten pairwise comparisons and adjusted the significance level using the Bonferroni correction. The adjusted significance level was set to $\alpha_{\text{adj}} = 0.05/10 = 0.005$. Only p-values below this threshold were considered statistically significant. Table 12 illustrates part of the results of paired t-tests with Bonferroni correction comparing the 'All' group with other groups (CentKNNCos, Cent, kNNCos, and kNNL2) on the Al Arabiya dataset. From Table A.1, we can easily conclude ties and losses, in line with results of Table V. Similar calculations are done for all datasets. In addition, for Table VI, the comparison is made in a similar fashion between the top-performing method "TF-IDF" and our two *Tasneef* variants, fastText, and AraVec baselines.

TABLE 12: Tie/Loss evaluation between 'All' DBMFs group and other DBMFs groups for AL Arabiya dataset.

Reference group	**Other DBMFs groups**	**p-value**	**Reject null hypothesis**	**Decision**
All	CentKNNCos	0.150703	False	Tie $\Leftrightarrow$
All	Cent	0.028436	False	Tie $\Leftrightarrow$
All	KNNCos	0.001092	True	Loss $\Downarrow$
All	KNNL2	0.000008	True	Loss $\Downarrow$

Bibliography

Ababneh, Ahmad Hussein (2022). "Investigating the relevance of Arabic text classification datasets based on supervised learning". In: *Journal of Electronic Science and Technology* 20.2, p. 100160. ISSN: 1674-862X. DOI: https://doi.org/10.1016/j.jnlest.2022.100160. URL: https://www.sciencedirect.com/science/article/pii/S1674862X22000131.

Abbas, Mourad and Kamel Smaïli (Sept. 2005). "Comparison of Topic Identification methods for Arabic Language". In: *International Conference on Recent Advances in Natural Language Processing - RANLP 2005*. 14-17. Borovets, Bulgaria. URL: https://inria.hal.science/inria-00000448.

Abbas, Mourad, Kamel Smaili, and Daoud Berkani (Oct. 2011). "Evaluation of Topic Identification Methods on Arabic Corpora". In: *Journal of Digital Information Management* 9, pp. 185–192.

Abd, Dhafar, Ahmed Sadiq, and Ayad Abbas (Jan. 2020). "Classifying Political Arabic Articles Using Support Vector Machine with Different Feature Extraction". In: pp. 79–94. ISBN: 978-3-030-38751-8. DOI: 10.1007/978-3-030-38752-5_7.

Abdallah, Tarek Amr and Beatriz de La Iglesia (2015). *Survey on Feature Selection*. arXiv: 1510.02892 [cs.LG]. URL: https://arxiv.org/abs/1510.02892.

Abdullah, Malak, Mirsad Hadzikadic, and Samira Shaikh (Dec. 2018). "SEDAT: Sentiment and Emotion Detection in Arabic Text Using CNN-LSTM Deep Learning". In: pp. 835–840. DOI: 10.1109/ICMLA.2018.00134.

Abozied, Ashraf (May 2019). "An effective dimension reduction algorithm for clustering Arabic text". In: *Egyptian Informatics Journal* 21. DOI: 10.1016/j.eij.2019.05.002.

Abu-Arqoub, Mohammad, Ghassan F. Issa, and Wael M. Hadi (2019). "The Impact of Feature Selection Methods for Classifying Arabic Textual Data". In: *International Journal of Recent Technology and Engineering (IJRTE)* 8.4, pp. 1062–1069. ISSN: 2277-3878.

Abuaiadah, Diab, Jihad El-Sana, and Walid Abusalah (Sept. 2014). "On the Impact of Dataset Characteristics on Arabic Document Classification". In: *International Journal of Computer Applications* 101, pp. 31–38. DOI: 10.5120/17701-8680.

Al-Anzi, Fawaz S. and Dia AbuZeina (2017). “Toward an enhanced Arabic text classification using cosine similarity and Latent Semantic Indexing”. In: *J. King Saud Univ. Comput. Inf. Sci.* 29, pp. 189–195. URL: https://api.semanticscholar.org/CorpusID:62377425.

Al-Azani, Sadam and El-Sayed M. El-Alfy (2017). “Using Word Embedding and Ensemble Learning for Highly Imbalanced Data Sentiment Analysis in Short Arabic Text”. In: *Procedia Computer Science* 109. 8th International Conference on Ambient Systems, Networks and Technologies, ANT-2017 and the 7th International Conference on Sustainable Energy Information Technology, SEIT 2017, 16-19 May 2017, Madeira, Portugal, pp. 359–366. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.05.365. URL: https://www.sciencedirect.com/science/article/pii/S1877050917310347.

Al khurayji, Raed and Ahmed Sameh (Nov. 2017). “An Effective Arabic Text Classification Approach Based on Kernel Naive Bayes Classifier”. In: *International Journal of Artificial Intelligence Applications* 8, pp. 01–10. DOI: 10.5121/ijaia.2017.8601.

Al Qadi, Leen et al. (Oct. 2019). “Arabic Text Classification of News Articles Using Classical Supervised Classifiers”. In: pp. 1–6. DOI: 10.1109/ICTCS.2019.8923073.

Al-Taani, A. T. and S. H. Al-Sayadi (2020). “Classification of Arabic Text Using Singular Value Decomposition and Fuzzy C-Means Algorithms”. In: *Applications of Machine Learning*. Ed. by P. Johri, J. K. Verma, and S. Paul. Singapore: Springer Singapore, pp. 111–123. DOI: 10.1007/978-981-15-3357-0_8.

Al-Tahrawi, Mayy M. and Sumaya N. Al-Khatib (2015). “Arabic text classification using Polynomial Networks”. In: *Journal of King Saud University - Computer and Information Sciences* 27.4, pp. 437–449. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2015.02.003. URL: https://www.sciencedirect.com/science/article/pii/S131915781500066X.

Al-thubaity, Abdulmohsen, Muneera Alhoshan, and Itisam Hazzaa (Jan. 2015). “Using Word N-Grams as Features in Arabic Text Classification”. In: vol. 569, pp. 35–43. ISBN: 978-3-319-10389-1. DOI: 10.1007/978-3-319-10389-1_3.

Alalyani, Nada and Souad Larabi Marie-Sainte (2018). “NADA: New Arabic Dataset for Text Classification”. In: *International Journal of Advanced Computer Science and Applications* 9.9. DOI: 10.14569/IJACSA.2018.090928. URL: http://dx.doi.org/10.14569/IJACSA.2018.090928.

Alghamdi, Nuha and F. Assiri (2019). “A Comparison of fastText Implementations Using Arabic Text Classification”. In: pp. 306–311. DOI: 10.1007/978-3-030-29513-4_21.

Alhawarat, Mohammad and Ahmad O. Aseeri (2020). “A Superior Arabic Text Categorization Deep Model (SATCDM)”. In: *IEEE Access* 8, pp. 24653–24661. URL: https://api.semanticscholar.org/CorpusID:211118148.

Alnemer, Khalid A. et al. (2015). “Are health-related tweets evidence based? Review and analysis of health-related tweets on Twitter”. In: *J. Med. Internet Res.* 17.10, e246.

Alsaleh, Deem and Souad Larabi Marie-Sainte (June 2021). “Arabic Text Classification Using Convolutional Neural Network and Genetic Algorithms”. In: *IEEE Access* PP, pp. 1–1. DOI: 10.1109/ACCESS.2021.3091376.

Alshaer, H. N., M. A. Otair, L. Abualigah, et al. (2021). “Feature selection method using improved CHI Square on Arabic text classifiers: analysis and application”. In: *Multimedia Tools and Applications* 80, pp. 10373–10390. DOI: 10.1007/s11042-020-10074-6. URL: https://doi.org/10.1007/s11042-020-10074-6.

Alshammari, Riyad (2018). “Arabic Text Categorization using Machine Learning Approaches”. In: *International Journal of Advanced Computer Science and Applications* 9.3. DOI: 10.14569/IJACSA.2018.090332. URL: http://dx.doi.org/10.14569/IJACSA.2018.090332.

Alwehaibi, Ali and Kaushik Roy (2018). “Comparison of Pre-Trained Word Vectors for Arabic Text Classification Using Deep Learning Approach”. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1471–1474. URL: https://api.semanticscholar.org/CorpusID:58674228.

Aronoff, Mark and Kirsten Fudeman (2016). *Morphology and Morphological Analysis*. 2nd ed. Blackwell Publishing. URL: https://books-library.net/files/books-library.online-12181715Je9T4.pdf.

Atef Mosa, Mohamed (Sept. 2022). “Feature selection based on ACO and knowledge graph for Arabic text classification”. In: *Journal of Experimental Theoretical Artificial Intelligence*. DOI: 10.1080/0952813X.2022.2125588.

Ayed, Rabii, M. Labidi, and M. Maraoui (2017). “Arabic text classification: New study”. In: *2017 International Conference on Engineering MIS (ICEMIS)*, pp. 1–7. DOI: 10.1109/ICEMIS.2017.8273037.

Azroumahli, Chaimae, Yacine Elyounoussi, and Hassan Badir (Nov. 2023). “BERT for Arabic NLP Applications: Pretraining and Finetuning MSA and Arabic Dialects”. In: pp. 60–72. ISBN: 978-3-031-47365-4. DOI: 10.1007/978-3-031-47366-1_5.

Baali, Massa and Nada Ghneim (Oct. 2019). “Emotion analysis of Arabic tweets using deep learning approach”. In: *Journal of Big Data* 6. DOI: 10.1186/s40537-019-0252-x.

Bahassine, Said et al. (2020). “Feature selection using an improved Chi-square for Arabic text classification”. In: *Journal of King Saud University - Computer and Information Sciences* 32.2, pp. 225–231. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2018.05.010. URL: https://www.sciencedirect.com/science/article/pii/S131915781730544X.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *CoRR* abs/1409.0473. URL: https://api.semanticscholar.org/CorpusID:11212020.

Bellman, Richard (1957). *Dynamic Programming*. Princeton University Press.

Biniz, Mohamed et al. (Sept. 2018). “Arabic Text Classification Using Deep Learning Technics”. In: *International Journal of Grid and Distributed Computing* 11, pp. 103–114. DOI: 10.14257/ijgdc.2018.11.9.09.

Boudad, Naaima et al. (July 2017). “Sentiment analysis in Arabic: A review of the literature”. In: *Ain Shams Engineering Journal* 9. DOI: 10.1016/j.asej.2017.04.007.

Canuto, Sérgio, Marcos André Gonçalves, and Fabrício Benevenuto (2016). “Exploiting New Sentiment-Based Meta-level Features for Effective Sentiment Analysis”. In: *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. WSDM ’16. San Francisco, California, USA: Association for Computing Machinery, 53–62. ISBN: 9781450337168. DOI: 10.1145/2835776.2835821. URL: https://doi.org/10.1145/2835776.2835821.

Canuto, Sérgio et al. (2018). “A Thorough Evaluation of Distance-Based Meta-Features for Automated Text Classification”. In: *IEEE Transactions on Knowledge and Data Engineering* 30.12, pp. 2242–2256. DOI: 10.1109/TKDE.2018.2820051.

Carnie, Andrew (2013). *Syntax: A Generative Introduction*. 3rd. Oxford, UK: Wiley-Blackwell.

Chantar, H. et al. (2019). “Feature selection using binary grey wolf optimizer with elite-based crossover for Arabic text classification”. In: *Neural Computing and Applications* 32, pp. 12201 –12220. DOI: 10.1007/s00521-019-04368-6.

Cheng, Weiwei and Eyke Hüllermeier (Sept. 2009). “Combining Instance-Based Learning and Logistic Regression for Multilabel Classification”. In: *Machine Learning* 76, pp. 211–225. DOI: 10.1007/s10994-009-5127-5.

Chung, Junyoung et al. (2014). “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. In: *ArXiv* abs/1412.3555. URL: https://api.semanticscholar.org/CorpusID:5201925.

Cover, T. and P. Hart (1967). “Nearest neighbor pattern classification”. In: *IEEE Transactions on Information Theory* 13.1, pp. 21–27. DOI: 10.1109/TIT.1967.1053964.

Cover, Thomas M. and Joy A. Thomas (2006). *Elements of Information Theory*. 2nd ed. John Wiley & Sons, Inc. ISBN: 9780471241959. DOI: 10.1002/047174882X.

Cunha, Washington et al. (2020). “Extended pre-processing pipeline for text classification: On the role of meta-feature representations, sparsification and selective sampling”. In: *Information Processing Management* 57.4, p. 102263. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2020.102263. URL: https://www.sciencedirect.com/science/article/pii/S030645731931461X.

Darwish, Kareem (June 2013). “Arabizi Detection and Conversion to Arabic”. In: *Proceedings of the 2014 Workshop on Language Technologies for the Arab World*. DOI: 10.3115/v1/W14-3629.

Devlin, Jacob et al. (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *North American Chapter of the Association for Computational Linguistics*. URL: https://api.semanticscholar.org/CorpusID:52967399.

Duwairi, Rehab (Aug. 2014). “Arabic Sentiment Analysis Using Supervised Classification”. In: DOI: 10.1109/FiCloud.2014.100.

Einea, Omar, Ashraf Elnagar, and Ridhwan Debsi (June 2019). “SANAD: Single-Label Arabic News Articles Dataset for Automatic Text Categorization”. In: *Data in Brief* 25, p. 104076. DOI: 10.1016/j.dib.2019.104076.

El-Alami, Fatima-Zahra and Said Ouatik El Alaoui (2016). “An Efficient Method based on Deep Learning Approach for Arabic Text Categorization”. In: URL: https://api.semanticscholar.org/CorpusID:149450447.

El-Alami, Fatima-Zahra, Said Ouatik El Alaoui, and Noureddine En-Nahnahi (2021). “Contextual semantic embeddings based on fine-tuned AraBERT model for Arabic text multi-class categorization”. In: *J. King Saud Univ. Comput. Inf. Sci.* 34, pp. 8422–8428. DOI: 10.1016/J.JKSUCI.2021.02.005.

El-Haj, Mo and Rim Koulali (Jan. 2013). “El-Haj, M., Koulali, R. "KALIMAT a Multipurpose Arabic Corpus" at the Second Workshop on Arabic Corpus Linguistics (WACL-2) 2013”. In.

Elbarougy, Reda, G.M. Behery, and Dr-Akram El Khatib (Jan. 2020). “A Proposed Natural Language Processing Preprocessing Procedures for Enhancing Arabic Text Summarization”. In: pp. 39–57. ISBN: 978-3-030-34613-3. DOI: 10.1007/978-3-030-34614-0_3.

Elhassan, Rasha Mamoun and Mahmoud Ali (2019). “The Impact of Feature Selection Methods for Classifying Arabic Texts”. In: *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, pp. 1–6. URL: https://api.semanticscholar.org/CorpusID:198931835.

Elisseeff, André and Jason Weston (2001). “A kernel method for multi-labelled classification”. In: *Advances in Neural Information Processing Systems*. Ed. by T. Dietterich, S. Becker, and Z. Ghahramani. Vol. 14. MIT Press. URL: https://proceedings.neurips.cc/paper_files/paper/2001/file/39dcaf7a053dc372fbc391d4e6b5d693-Paper.pdf.

Elnagar, Ashraf, Ridhwan Al-Debsi, and Omar Einea (2020). “Arabic text classification using deep learning models”. In: *Information Processing Management* 57.1, p. 102121. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2019.102121. URL: https://www.sciencedirect.com/science/article/pii/S0306457319303413.

Essma, Selab and Ahmed Guessoum (Jan. 2015). “Building TALAA, a Free General and Categorized Arabic Corpus”. In: vol. 1. DOI: 10.5220/0005352102840291.

Fan, Rong-En et al. (Aug. 2008). “LIBLINEAR: a library for large linear classification”. In: *Journal of Machine Learning Research* 9, pp. 1871–1874. DOI: 10.1145/1390681.1442794.

Farghaly, Ali et al. (Jan. 2009). “Arabic Natural Language Processing: Challenges and Solutions”. In: *ACM Transactions on Asian Language Information Processing (TALIP)* 8.

Ferguson, Charles A. (1972). “Diglossia”. In: *Language and Social Context*. Ed. by Pier Paolo Gigliolo. Originally published in 1959, pp. 232–251.

Ghojogh, Benyamin et al. (2019). *Feature Selection and Feature Extraction in Pattern Analysis: A Literature Review*. arXiv: 1905.02845 [cs.LG]. URL: https://arxiv.org/abs/1905.02845.

Gholitabar, Marzieh and Atiyeh Damavandi Kamali (2015). “The Quran and the Development of Arabic Linguistics”. In: URL: https://api.semanticscholar.org/CorpusID:197616773.

Goldberg, Yoav (2017). *Neural Network Methods for Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Cham, Switzerland: Springer. DOI: 10.1007/978-3-031-02165-7.

Gopal, Siddharth and Yiming Yang (2010). “Multilabel classification with meta-level features”. In: *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’10. Geneva, Switzerland: Association for Computing Machinery, 315–322. ISBN: 9781450301534. DOI: 10.1145/

1835449.1835503. URL: https://doi.org/10.1145/1835449.1835503.

Habash, Nizar Y. (2010). *Introduction to Arabic Natural Language Processing*. 1st ed. Springer Cham. DOI: 10.1007/978-3-031-02139-8.

Hadni, Meryeme and Hassane Hjiaj (2023). "An Improved Chaotic Sine Cosine Firefly Algorithm for Arabic Feature Selection". In: *Proceedings of the 6th International Conference on Big Data and Internet of Things*. Ed. by Mohamed Lazaar et al. Cham: Springer International Publishing, pp. 84–94.

Harrag, Fouzi and Eyas Al-Qawasmah (Aug. 2010). "Improving Arabic Text Categorization Using Neural Network with SVD." In: *JDIM* 8, pp. 233–239.

Haynes, Winston (2013). "Bonferroni Correction". In: *Encyclopedia of Systems Biology*. Ed. by Werner Dubitzky et al. New York, NY: Springer New York, pp. 154–154. ISBN: 978-1-4419-9863-7. DOI: 10.1007/978-1-4419-9863-7_1213. URL: https://doi.org/10.1007/978-1-4419-9863-7_1213.

Hochreiter, Sepp and Jürgen Schmidhuber (Dec. 1997). "Long Short-term Memory". In: *Neural computation* 9, pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

Jolliffe, I. T. (2011). *Principal Component Analysis*. 2nd. Springer.

Jolliffe, Ian (Jan. 2002). "Principal Component Analysis Springer Verlag". In.

Jurafsky, Daniel and James H. Martin (2023). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 3rd. Comments and typos welcome! Draft of January 7, 2023.

Kathrein, Abu Kwaik et al. (Nov. 2018). "A Lexical Distance Study of Arabic Dialects". In: *Procedia Computer Science* 142, pp. 2–13. DOI: 10.1016/j.procs.2018.10.456.

Kowsari, Kamran et al. (Apr. 2019). "Text Classification Algorithms: A Survey". In: *Information (Switzerland)* 10. DOI: 10.3390/info10040150.

Kyriakopoulou, Tonia and Theodore Kalamboukis (July 2007). "Using clustering to enhance text classification". In: pp. 805–806. DOI: 10.1145/1277741.1277918.

LeCun, Yann et al. (1989). "Handwritten Digit Recognition with a Back-Propagation Network". In: *Neural Information Processing Systems*. URL: https://api.semanticscholar.org/CorpusID:2542741.

Louail, Maroua and Chafia Kara-Mohamed (2023). "Distance-Based Meta-Features for Arabic Text Classification". In: *2023 Intelligent Methods, Systems, and Applications (IMSA)*, pp. 257–262. URL: https://api.semanticscholar.org/CorpusID:261127120.

Louail, Maroua, Chafia Kara-Mohamed, and Aboubekeur Hamdi-Cherif (2024). "Tasneef: A Fast and Effective Hybrid Representation Approach for Arabic Text Classification". In: *IEEE Access* 12, pp. 120804–120826. DOI: 10.1109/ACCESS.2024.3450507.

Louail, Maroua, Chafia Kara-Mohamed Hamdi-Cherif, and Aboubekeur Hamdi-Cherif (2021). "Arabic text classification using principal component analysis with different supervised classifiers". In: *2021 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, pp. 1–6. DOI: 10.1109/ICECET52533.2021.9698799.

Lulu, Leena and Ashraf Elnagar (2018). "Automatic Arabic Dialect Classification Using Deep Learning Models". In: *Procedia Computer Science* 142. Arabic Computational Linguistics, pp. 262–269. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.10.489. URL: https://www.sciencedirect.com/science/article/pii/S1877050918321938.

Mahmoud, Adnen and Mounir Zrigui (2019). "Deep Neural Network Models for Paraphrased Text Classification in the Arabic Language". In: *International Conference on Applications of Natural Language to Data Bases*. URL: https://api.semanticscholar.org/CorpusID:195353278.

Mikolov, Tomas et al. (2013). "Efficient Estimation of Word Representations in Vector Space". In: *International Conference on Learning Representations*. URL: https://api.semanticscholar.org/CorpusID:5959482.

Mikolov, Tomas et al. (Dec. 2017). "Advances in Pre-Training Distributed Word Representations". In.

Mohammed, Ammar and Rania Kora (Sept. 2019). "Deep learning approaches for Arabic sentiment analysis". In: *Social Network Analysis and Mining* 9. DOI: 10.1007/s13278-019-0596-4.

Obeid, Ossama et al. (2020). "CAMeL Tools: An Open Source Python Toolkit for Arabic Natural Language Processing". In: *International Conference on Language Resources and Evaluation*. URL: https://api.semanticscholar.org/CorpusID:218973981.

Oguike, Osondu (2021). *A First Course in Artificial Intelligence*. Western Binding. Bentham Science Publishers, p. 340. ISBN: 9781681088549, 1681088541.

Pang, Guansong, Huidong Jin, and Shengyi Jiang (2015). "CenKNN: A scalable and effective text classifier". English. In: *Data Mining and Knowledge Discovery* 29.3, pp. 593–625. ISSN: 1384-5810. DOI: 10.1007/s10618-014-0358-x.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Conference on Empirical Methods in Natural Language Processing*. URL: https://api.semanticscholar.org/CorpusID:1957433.

Qadi, Leen Al et al. (2019). "Arabic Text Classification of News Articles Using Classical Supervised Classifiers". In: *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, pp. 1–6. DOI: 10.1109/ICTCS.2019.8923073.

Quinlan, J. R. (1986). "Induction of Decision Trees". In: *Machine Learning* 1, pp. 81–106. DOI: 10.1007/BF00116251.

Radford, Alec and Karthik Narasimhan (2018). *Improving Language Understanding by Generative Pre-Training*. Tech. rep. OpenAI. URL: https://api.semanticscholar.org/CorpusID:49313245.

Raj, Hans et al. (2018). "LSTM Based Short Message Service (SMS) Modeling for Spam Classification". In: *Proceedings of the 2018 International Conference on Machine Learning Technologies*. ICMLT '18. Jinan, China: Association for Computing Machinery. ISBN: 9781450364324. DOI: 10.1145/3231884.3231895. URL: https://doi-org.sndl1.arn.dz/10.1145/3231884.3231895.

Redkin, Oleg and Olga Bernikova (2016). "Globalization and the Arabic language acquisition". In: *Proceedings of The 20th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI)*, pp. 196–199.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning representations by back-propagating errors". In: *Nature* 323, pp. 533–536. URL: https://api.semanticscholar.org/CorpusID:205001834.

Ryding, Karin C. (2014). "Arabic inflectional morphology". In: *Arabic: A Linguistic Introduction*. Cambridge University Press, 89–106.

Saad, Motaz and Wesam Ashour (Nov. 2010). "OSAC: Open Source Arabic Corpora". In: DOI: 10.13140/2.1.4664.9288.

Sabri, Tarik, Omar El Beggar, and Mohamed Kissi (Feb. 2024). "An improved Arabic text classification method using word embedding". In: *International Journal of Electrical and Computer Engineering (IJECE)* 14, pp. 721–731. DOI: 10.11591/ijece.v14i1.pp721-731.

Sagheer, Dania and Fadel Sukkar (July 2018). "Arabic Sentences Classification via Deep Learning". In: *International Journal of Computer Applications* 182, pp. 40–46. DOI: 10.5120/ijca2018917555.

Schuster, Mike and Kuldip K. Paliwal (1997). “Bidirectional recurrent neural networks”. In: *IEEE Trans. Signal Process.* 45, pp. 2673–2681. URL: https://api.semanticscholar.org/CorpusID:18375389.

Siino, Marco, Ilenia Tinnirello, and Marco La Cascia (2024). “Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers”. In: *Information Systems* 121, p. 102342. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2023.102342. URL: https://www.sciencedirect.com/science/article/pii/S0306437923001783.

Silla, Carlos Nascimento and Alex Alves Freitas (2010). “A survey of hierarchical classification across different application domains”. In: *Data Mining and Knowledge Discovery* 22, pp. 31–72. URL: https://api.semanticscholar.org/CorpusID:207113055.

Skillicorn, David (Jan. 2012). *Understanding High-Dimensional Spaces*. ISBN: 978-3-642-33397-2. DOI: 10.1007/978-3-642-33398-9.

Sokolova, Marina and Guy Lapalme (July 2009). “A systematic analysis of performance measures for classification tasks”. In: *Information Processing & Management* 45.4, pp. 427–437. DOI: 10.1016/j.ipm.2009.03.002.

Soliman, Abu Bakr, Kareem Eissa, and Samhaa R. El-Beltagy (2017). “AraVec: A set of Arabic Word Embedding Models for use in Arabic NLP”. In: *Procedia Computer Science* 117. Arabic Computational Linguistics, pp. 256–265. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.10.117. URL: https://www.sciencedirect.com/science/article/pii/S1877050917321749.

Spärck Jones, Karen (Dec. 1972). “Jones, K.S.: A Statistical Interpretation of Term Specificity and its Application in Retrieval. Journal of Documentation 28(1), 11-21”. In: *Journal of Documentation* 28, pp. 11–21. DOI: 10.1108/eb026526.

Sugiyama, M. (2007). “Dimensionality Reduction of Multimodal Labeled Data by Local Fisher Discriminant Analysis”. In: *Journal of Machine Learning Research* 8, pp. 1027–1061.

Sundus, Katrina, Fatima Al-Haj, and Bassam Hammo (2019). “A Deep Learning Approach for Arabic Text Classification”. In: *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, pp. 1–7. DOI: 10.1109/ICTCS.2019.8923083.

Tsai, C. F. et al. (2011). “Distance-based features in pattern classification”. In: *EURASIP Journal on Advances in Signal Processing* 2011.62, pp. 1–12.

Vapnik, Vladimir, Steven E. Golowich, and Alex Smola (1996). “Support vector method for function approximation, regression estimation and signal processing”. In: *Proceedings of* *the 9th International Conference on Neural Information Processing Systems*. NIPS'96. Denver, Colorado: MIT Press, 281–287.

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Neural Information Processing Systems*. URL: https://api.semanticscholar.org/CorpusID:13756489.

Versteegh, Kees (2014). *The Arabic Language*. 2nd ed. Edinburgh University Press.

Wahbeh, Abdullah and Mohammed Al-Kabi (Jan. 2012). "Comparative Assessment of the Performance of Three WEKA Text Classifiers Applied to Arabic Text". In: *ABHATH AL-YARMOUK: Basic Science Engineering* 21, pp. 15–28.

Wahdan, Ahlam, Mostafa Al-Emran, and Khaled Shaalan (May 2023). "A systematic review of Arabic text classification: areas, applications, and future directions". In: *Soft Computing* 28, pp. 1–22. DOI: 10.1007/s00500-023-08384-6.

Yang, Yiming and Jan O. Pedersen (1997). "A Comparative Study on Feature Selection in Text Categorization". In: *International Conference on Machine Learning*. URL: https://api.semanticscholar.org/CorpusID:5083193.

Zahedi, M and A Ghanbari Sorkhi (2013). "Improving text classification performance using PCA and recall-precision criteria". In: *Arabian Journal for Science and Engineering* 38, pp. 2095–2102.

Zerrouki, Taha (2010). *Arabic Stop Words*. URL: https://github.com/linuxscout/arabicstopwords.

— (2023). "PyArabic: A Python package for Arabic text". In: *Journal of Open Source Software* 8.84, p. 4886. DOI: 10.21105/joss.04886.

Zhang, Min-Ling and Zhi-Hua Zhou (July 2007). "ML-KNN: A lazy learning approach to multi-label learning". In: *Pattern Recogn.* 40.7, 2038–2048. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2006.12.019. URL: https://doi.org/10.1016/j.patcog.2006.12.019.