ISSN 2278-3091
Volume 10, No.1, January - February 2021
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse061012021.pdf
https://doi.org/10.30534/ijatcse/2021/061012021

The effects of Pre-Processing Techniques on Arabic Text Classification

Anoual El Kah[1], Imad Zeroual[2]
[1]Faculty of Sciences, Mohammed First University, Morocco, elkah.anoual.mri@gmail.com
[2] L-STI, T-IDMS, Faculty of Sciences and Techniques, Moulay Ismail University, Morocco, mr.imadine@gmail.com

ABSTRACT

In the last two decades, the amount of available Arabic text data on the World Wide Web is dramatically growing, making it the fourth most used language on the web. Accordingly, the demand for efficient Arabic text classification is increasing, especially for web page content filtering, information retrieval, and e-mail spam detection. Several Machine Learning algorithms have been implemented to classify Arabic documents. However, the results achieved are not comparable with those obtained in other languages such as English, primarily when using preprocessing techniques that do not take into consideration the Arabic language features. This paper investigates the impact of wisely selected preprocessing techniques on the efficiency of different text classification algorithms. The effects of stop words removal, stemming, lemmatization, and all possible combinations are examined. The reported results (+10.75% to +28.73%) prove the effectiveness of using these techniques either individually or in combination.

Key words: Arabic text classification, Lemmatization, Stemming, Stop words removal, Text preprocessing.

1. INTRODUCTION

There have been a lot of efforts and studies were devoted to Arabic natural language processing and its applications [1]. Last years have witnessed remarkable progress in building Arabic corpora [2] and developing robust morphological analyzers [3], which paved the way for highly data-driven approaches like text classification, information retrieval, and machine translation.

In terms of Arabic text classification, several Machine Learning algorithms have been successfully implemented such as Naïve Bayes (NB), Support Vector Machine (SVM), Decision Tree (DT), K-Nearest Neighbors (K-NN), Artificial Neural Network (ANN) [4]. However, for the Arabic language, the text classifier performance is not only influenced by the algorithm implemented; also, the nature of the language has a great impact on the developed classifier. Therefore, most scholars have often involved some text preprocessing techniques that can deal effectively with the richness morphology and lexicon of such language. Moreover, deriving text preprocessing techniques from western scholars, even if they were the forerunner in natural language processing, is a drawback for a morphologically complex language such as Arabic. Expressly, the morphology of Arabic differs from western languages (e.g., English), and adopting such techniques without considering the language features will lead to different results than expected.

Text preprocessing techniques are normally used to reduce the document size, facilitate feature selection, and increase the processing speed. Regarding the text classification, the main purpose of involving a text preprocessing task is dealing with the problem of the high dimensionality of data. In other words, we need to reduce as much as possible the size of text features without leading to a system's deterioration. To overcome this issue, several techniques have been proposed. These include, among others, stop words elimination, stemming, and lemmatization.

Stop words elimination aims to remove redundant words that carry no significant information or indicate the subject of the processed document. Whereas, stemming and lemmatization are dedicated to regroup the words that are morphologically and semantically related.

In this paper, we will demonstrate that selecting the right text preprocessing techniques may lead to positive outcomes in Arabic text classification. To this end, we deeply examined the contributions of common techniques, namely stop words removal, stemming, and lemmatization to classification accuracy of three well-known algorithms, Naïve Bayesian, Support Vector Machines, and Decision Trees.

After this introduction, the main content of the paper is structured as follows: Section 2 presents the challenges faced

by text classifiers due to particular Arabic language characteristics. Next, the corpus we compiled exclusively for this study is presented in Section 3. Section 4 introduces different preprocessing techniques that we implemented. Section 5 exposes the feature extraction and selection methods used for reduction the data dimensionality. Furthermore, three classifiers, that are commonly implemented to automatically classify documents, are presented. Finally, the results of the experimental investigation are illustrated. In Section 6, conclusions and future directions are constituting the final section.

2. ARABIC TEXT CLASSIFICATION CHALLENGES

Dealing with Arabic language features and implementing the right text preprocessing techniques always have, and still are, intriguing researchers in the field of Arabic text classification.

Arabic is an old Semitic language and its morphology is deeply rooted and well established a long time ago, which makes it one of the most highly inflected languages. Thus, a word can represent a whole sentence through sequential concatenation. For instance, the Arabic word "أَفَاسْتَسْقَيْنَاكُمُوهَا", which contains 15 letters and 10 diacritics, means in English "Then did we asked you for it to drink". Besides, according to a study [5] that investigated the average length of Arabic words in news articles using a corpus of one billion words, 75% of the words have a length above six letters. Moreover, Arabic has very rich lexicon of synonyms. E.g., it is believed that the word "lion" in Arabic has between 350 to 500 synonyms[1]. As a result, it is recommended to involve preprocessing techniques that aim to deal with such language features.

Since the aim of the preprocessing techniques is to minimize information loss while maximizing reduction in data dimensionality, stop words elimination is then recommended in most cases. However, random removal of stop words may deteriorate the text classifier performance. Therefore, a stop words list must be wisely chosen, because the purpose for which it is generated influences the performance of the classifier.

An interested survey [6], that covered 17 studies that investigated the stemming impact on Arabic text classification performance, reported that nine experiments proved that there is an enhancement if stemming was performed; whereas, eight experiments claimed the opposite. Not surprisingly, it looks like we are in front of a debate regarding the impact of stemming. According to the authors, this may return to the stemming algorithm implemented. Note that, there are two main types of Arabic stemmers, the light stemmer and the root-based/heavy stemmer. The light stemmer aims to remove clitics/affixes without trying to find roots (e.g., [7]). The heavy stemmer reduces inflected words to their roots (e.g., [8]). Thus, if a root-based stemming is performed, information like the grammatical features, the part of speech, and the meaning of all the words will be lost. For example, the words "عَيْن" (i.e., eye), "مَعَانِي" (i.e., meanings), "أُعِينُ" (i.e., I help), "عُيُون" (i.e., fountains) are related to the same root "عين". Consequently, the preprocessing techniques that are based totally or partially on root-based stemming will experience a deterioration in performance; otherwise, no improvement will be assumed (e.g., [9]). It is worth mentioning that the reason why a root-based stemmer is used may return to the fact that is very efficient in feature vector dimension reduction.

Typically, lemmatization reduces indexing data dimension more than stemming, while still enhancing the text classification performance [10]. The reason is that a lemmatizer regroups semantically related words although they are morphologically different from each other. For instance, the lemma of the words "كُتُب" (i.e., books), "كُتَيِّبَات" (i.e., manuals), and "كِتَابَان" (i.e., two books) is "كِتَاب" (i.e., book). In comparison to the stemming techniques, the light stemmer cannot relate the three given words to the same stem; instead, each word will be considered as an independent stem. On the contrary, the heavy stemmer will relate the three words to the same root "كتب". However, if we add the word "كَتِيبَة" (i.e., troop) to the text, the heavy stemmer still relates this new word to the same previous root "كتب" although the word troop has different meaning from the other words (books and manuals). Whereas, the lemmatizer will relate it to another lemma that shares the same meaning.

3. DATASETS

The World Wide Web becomes a vital source of digital documents due to the tremendous growth of its content primarily in news websites and social media. Since the Arabic language is currently the fourth most used language on the web [2], compiling Arabic datasets is more manageable especially with the availability of free web crawlers that make web scraping easier and accessible to everyone. These include, among others, HTTRACK[3], Heritrix[4], and Scrapy[5].

In order to build our corpus, we crawled different Arabic websites around the world using the HTTRACK web crawler. Then, a cleaning and normalizing process were performed to

[1] https://ar.wikipedia.org/wiki/قائمة_أسماء_الأسد_في_اللغة_العربية

[2] https://www.internetworldstats.com/stats7.htm
[3] http://www.httrack.com/
[4] https://github.com/internetarchive/heritrix3/wiki
[5] https://scrapy.org/

keep only Arabic text. Since each website has different categories, we decided to build category-based datasets rather than website-based. Thus, we selected the most common categories namely politics, culture, economy, sport, health, and technology.

A published study [11] stated that the length of texts, as well as the number of articles in a given category, have an influence in Arabic news texts classification. Besides, our goal is to make the overall corpus representative and balance. To this end, we decided to collect for each category 50,000 articles. The final corpus includes a total of 300,000 articles, containing over 153 million words. In addition, each article consists of an average of 512 words, with a minimum of 287 words and a maximum of 737 words.

The criteria adopted to select the crawled websites are simples. First, over 40 news websites were selected based on their popularity in their region or country. Then, each website was reviewed if it includes at least the six categories that we mentioned earlier. Consequently, only nine websites were crawled according to these criteria. Table 1 present the list of crawled websites.

Table 1: List of crawled websites

Region/Country	Website
United Nations	www.news.un.org/ar/
Middle-east	www.aljazeera.net
UK	www.bbc.com/arabic
USA	www.arabic.cnn.com
Russia	www.arabic.rt.com
Germany	www.dw.com/ar/
Morocco	www.marocpress.com
Tunisia	www.turess.com
Iran	www.alalamtv.net/

4. TEXT PRE-PROCESSING

This section describes the preprocessing methods that we applied on the compiled datasets. The preprocessing methods assessed are stop words elimination, stemming, and lemmatization.

4.1 Stop Words Elimination

A stop word is a term that frequently appears in a text and carries no significant information or indicates the subject of the processed text. Compiling a stop words list mainly rely on two techniques. The first one is a rule-based technique that involves the use of morphological analysis (e.g., [12]). The generated list is a domain-independent list which is devoted for a general use. The second technique is a statistical approach that consists of using a frequency feature of a particular corpus (e.g., [13]). This is often the case when we have to generate a domain-dependent list that is used for a specific field.

Stop word elimination plays a major role in the pre-processing stage of text classification as well as other text processing applications. For instance, information retrieval [14], text summarization [15], and automatic translation [16].

Note that, the stop words elimination in the text classification context doesn't apply to consider only the particles; there are nouns and verbs which are also considered as stop-words. Besides, reference [17] claimed that random removal of stop words may significantly deteriorate text classification performance and lead to different results than expected.

Since our purpose is achieving a higher preprocessing efficiency without affecting the text classification performance, our approach consists to build a stop words list for a general use that includes all basic stop words and its inflected forms.

The basic stop words were collected from previously published lists. After reviewing and filtrating the compiled lists, we created a new list of roughly 1,000 domain-independent stop words. Then, we generated their inflected forms following a proposed technique that involves 123 Arabic clitics [18]. As a result, the final list comprises 11,403 stop words.

4.2 Stemming

In the case of the Arabic language, stemming has been the subject of several studies that have shown its effectiveness in both text classification and clustering [4], [6], [19], [20]. However, three recent comparative studies investigated different Arabic stemmers in the field of Arabic text classification. The first one [21] reported that the classification was more efficient with Tashaphyne light based stemmer[6] followed by Farasa [22], Khoja stemmer [8], Light10 [7], and finally Al Khalil Morph Sys [23]. What's more, this study found that the stemmer accuracy does not have an impact on the classifier efficiency in topic identification. The second study [24] focused on the impact of stemming techniques namely Information Science Research Institute (ISRI) [25], Tashaphyne, and ARLStem v1.0 [26] on Arabic document classification. Findings of this paper indicated that the ARLStem v1.0 outperforms the ISRI and Tashaphyne stemmers. The third study [27] claimed that the original ARLSTem v1.0 achieved the best result over different stemmers in Arabic text classification. The overall ranking for the rest of the stemmers came as follows: ARLSTem v1.1, Light10, Assem's stemmer[7], ISRI stemmer, and Soori's stemmer [28].

[6] https://pypi.org/project/Tashaphyne/ [last accessed: January 24, 2021]
[7] https://arabicstemmer.com/ [last accessed: January 24, 2021]

All in all, the three previous studies covered 10 Arabic stemmers and investigated their performances when applied in the field of Arabic text classification. As reported, the ARLSTem v1.0[8] is the best stemmer that demonstrated how its use may result positive outcomes in Arabic text classification. Therefore, it is the one implemented in our study.

ARLSTem is a light stemmer that is based on an algorithm of six tasks. The first three tasks are normalization, prefixes removal, and suffixes removal. The fourth and the fifth tasks are dedicated to stem nouns by transforming the plural to singular and the feminine to masculine. The sixth task is devoted to stem conjugated verbs.

4.3 Lemmatization

If text stemming regroups the words that are morphologically related; then, text lemmatization aims to regroup semantically related words. Unfortunately, involving the lemmatization as a preprocessing task in Arabic text classification is fairly limited. Maybe the raison is that the lemmatization is a complex level of text processing and most Arabic lemmatizers are proprietary and not publicly available compared to Arabic stemmers. However, several studies reported that using lemmatization is found to be efficient, in particular, for information retrieval [29], text summarization systems [30], and text indexation [31].

Generally, as much as the meaning of document content is well represented, its classification will be easier. Besides, lemmatization is regrouping semantically equivalent words that are written in different syntactic forms and relates them to their canonical base representation called lemma (i.e., a dictionary lookup form). Thus, involving lemmatization as a preprocessing task for text classification supposed to be quite advantageous.

As many other studies dedicated to Arabic text classification, the datasets we collected are from news websites, meaning they are written in the modern Arabic language. Therefore, the selected lemmatizer must successfully deal with such a language and hopefully be trained on a large amount of Arabic news articles. To this end, the lemmatizer of Madamira v2.1 [32] was used.

5. EXPERIMENTAL WORK

Basically, text classification systems followed the same methodology, which consists of collecting a corpus or datasets, applying preprocessing techniques, feature extraction (i.e., text representation), feature selection, and finally conducts the classification task. Since the focus of this paper is investigating the impact of the preprocessing phase on the whole classification procedure, the selection of algorithms used in subsequent phases was based on the recommendations of relevant prior works. In the sequel, feature extraction, feature selection, and classification tasks, are performed using WEKA (Waikato Environment for Knowledge Analysis) [33].

5.1 Feature extraction

Feature extraction is a task concerning the transformation of raw data into suitable inputs (i.e., features) that can be consumed by a particular Machine Learning algorithm. Expressly, the extracted features must represent the primary textual content in a format that will best fit the needs of the selected classifier algorithm. Except for deep learning neural networks, which can perform feature extraction by themselves, a minimum of feature extraction is always needed. Moreover, it is believed that a poor classifier fed with meaningful features may perform better than a robust classifier fed with low-quality features.

Regarding Arabic text classification, Bag-of-Words (BoW), Bag-of-Concepts (BoC), and Term Frequency–Inverse Document Frequency (TF-IDF) are the common techniques used for feature extraction [34], [35]. In this study, the TF-IDF is applied.

BoW simply generates a set of vectors containing the count of word occurrences in a given document. On the other hand, the TF-IDF associates each word in a document with a number that represents how relevant this word is in that document. Then, each document will contain information on the more important words and the less important ones as well. Consequently, documents with similar relevant words will have similar vectors. Although, both BoW and TF-IDF have been popular in their regard, TF-IDF usually performs better in Arabic text classification.

5.2 Feature selection

Feature selection or attribute selection is the natural successor task of feature extraction. It aims to identify a subset of the most significant features from the sparse feature space without affecting the classifier performance [36]. A quite number of comprehensive investigations and reviews on feature selection approaches that are explicitly designed for text classification have been published (e.g., [37]). In general, there are four feature selection approaches, namely filter method (e.g., [38]), wrapper method (e.g., [39]), embedded method (e.g., [40]), and hybrid method (e.g., [41]). Each one of these approaches has its pros and cons. These methods have strong theoretical foundations and have proved their superiority in feature selection. However, most feature selection approaches for text classification belong to the filter-based method due to its simplicity and efficiency. Chi-squared test (χ^2 test) is a filter method that is computationally fast, simple and has the ability to deal with a

[8] https://www.nltk.org/_modules/nltk/stem/arlstem.html [last accessed: January 24, 2021]

large dimensional feature. Chi-square proved its efficiency especially when applied with TF-IDF extracted features [42].

After feature extraction is implemented on the training datasets by calculating the TF-IDF score for each feature, Chi-square is performed for the feature selection. Expressly, the Chi-squared test is used for testing the independence between the occurrence of a specific feature and the occurrence of a specific category. The null hypothesis of the Chi-Squared test means that no relationship exists between them; i.e., they are independent. Next, we ranked the features by their score. Only top-rank features are then selected to serve as inputs for the next phase, the classification phase.

5.3 Classifiers

Many standard classifiers have been designed to automatically classify documents. In this section, we will focus on the following models: Naïve Bayesian algorithm (NB), Support Vector Machines (SVMs), and Decision Trees (J48). It is worth mentioning that the purpose of this phase is not to determine the best classifiers for Arabic documents classification but to investigate the effectiveness of the selected preprocessing techniques on the performance of the classifier.

The NB classifier assumption is that the probability of each input feature appearing in a given document is independent of the occurrence of another feature in the same document. Further, NB computes the probability that a particular document belongs to each of the categories with which the system has trained; then, assigns this document to the specific category with the highest probability. Since NB can output a probability for each possible category, it is possible to identify multiple categories to which a document may belong.

SVMs determines the best decision boundary between features that belong, and not belong as well, to a given category. It is stated by [43] that SVMs are not affected by the high dimensionality of the feature space, meaning they can manage all the features even if no feature selection techniques are performed. Consequently, SVMs are well suited for classification problems with dense concepts and sparse instances.

DT classifier is used to rebuild the pre-classified training dataset by constructing well-defined true/false-queries in the form of a tree structure. In this tree, the internal nodes are labeled by features, leaves represent the categories of documents, and branches represent conjunctions of features that lead to those categories. Given a document to classify, the constructed decision tree is used to predict which category the document should belong to. However, DT classifiers suffer badly in high dimensional feature spaces.

5.4 Results and discussion

As mentioned in the previous sections, we have compiled a corpus by selecting and crawling nine relevant news websites. This corpus includes category-based datasets that cover six categories, namely politics, culture, economy, sport, health, and technology. Each category comprises 50,000 articles. Each article consists of an average of 512 words, with a minimum of 287 words and a maximum of 737 words.

After cleaning and preparing the corpus, three different preprocessing techniques were performed, Stop Words (SWs) elimination, stemming, and lemmatization. Then, TF-IDF and Chi-square were implemented for feature extraction and feature selection, respectively. Finally, three well-known algorithms (NB, SVM, and DT J48) were used for the classification. Note that, the 10-fold cross-validation were used for the evaluation of performance accuracy.

During the experiments, the three preprocessing techniques and all their possible combinations were considered: SW removal, stemming, lemmatization, SWs removal and stemming, SWs removal and lemmatization, stemming and lemmatization, and finally, all the three techniques combined. Needless to say, the effectiveness of these techniques is compared with the case when none of these techniques is used. Table 2 exhibits the accuracies achieved by the classifiers without using any preprocessing technique and after involving each preprocessing technique and the possible combinations.

Table 2: 10 Folds cross-validation scores for evaluating preprocessing effectiveness on the classifiers

Preprocessing techniques	**10 Folds cross-validation scores (%)**		
	NB	**SVM**	**DT J48**
Without preprocessing	64.68	71.15	57.17
SW removal	81.30	87.90	74.70
Stemming	74.71	81.05	69.50
Lemmatization	79.29	83.42	69.83
SWs removal & Stemming	90.88	90.76	86.85
SWs removal & Lemmatization	92.48	92.75	89.29
Stemming & Lemmatization	80.95	85.42	73.33
All techniques	93.47	94.91	90.81

For more illustration, Figure 1 depicts the percentages of the improvement recorded for each classifier (i.e., BN, SVM, and DT J48) when the preprocessing techniques and their combinations were involved.

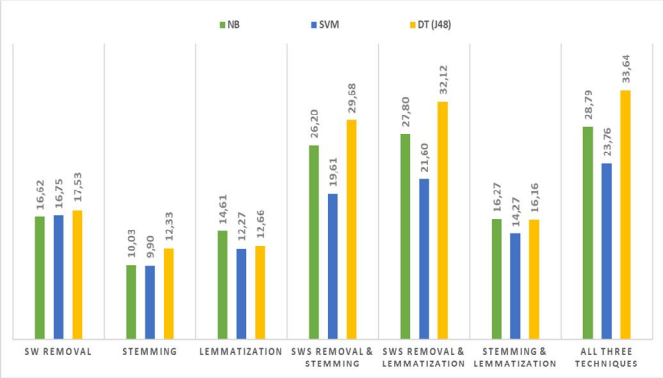

Figure 1: Improvement recorded in the classifiers when preprocessing techniques were involved

The results obtained prove the effectiveness of the preprocessing techniques selected (+10.75% to +16.97%). What's more, combining more techniques leads to better improvement than only individual technique does (+15.57% to +28.73%).

In this experiment, we observe that the average impact of the SWs removal (+16,97%) on the three classifiers performance is greater than those of stemming (+10,75%) and lemmatization (+13,18%). Similarly, the combinations that include SWs removal -i.e., SWs removal with stemming (an average of +25,16%) and SWs removal with lemmatization (an average of+27,17%)- perform better than the one that does not include SWs removal -i.e., stemming and lemmatization (an average of +15,57%)-. In fact, applying exclusively SWs removal (+16,97%) is more useful than using stemming and lemmatization combined (+15,57%). This implies that involving SWs removal is highly recommended especially if the classifier is based on a decision tree algorithm. A simple computation was conducted on our corpus shows that stop words represent 35% to 43% of the document content. Thus, removing these stop words maximize the reduction in data dimensionality, which enhances classifiers that suffer badly in high dimensional feature spaces such as DT J48.

On the other hand, when comparing stemming to lemmatization effects, the results show that this latter is slightly more beneficial. Expressly, the lemmatization enhances the classification by an average of +13.18%; whereas, +10.75% is the enhancement average recorded when stemming is involved. Likewise, the combination, which includes lemmatization and SWs removal, passes the combination of stemming and SWs removal by (+2.01%). This can be explained by the similarities found in the results obtained from the selected light stemmer (ARLSTem) and lemmatizer (Madamira lemmatizer). The most similarities are found between verbs stems and lemmas.

Finally, all the classifiers achieved the best results when all the three preprocessing techniques were involved (an average of +28.73%). Note that, the SVMs are the most accurate classifier in most cases, followed by NB and DT (J48).

6. CONCLUSION

This paper investigates the impact of widely used preprocessing techniques including stop words elimination, stemming, and lemmatization. Besides, all possible combinations of those preprocessing techniques are considered. To this end, we compiled a new balanced and large corpus with 300,000 articles which are equally distributed into six categories. Additionally, a new stop words list was generated. Based on our state-of-the-art review, robust developed tools were used for stemming and lemmatization. Similarly, feature extraction, feature selection, and text classification algorithms were selected based on previously published works.

The performed experiments confirm that well-selected preprocessing techniques have a great impact on Arabic text classification. Stop words removal has been shown to be the most beneficial techniques, especially for classification algorithms that suffer badly in high dimensional feature spaces like decision trees. However, the usability of the proposed preprocessing techniques, either individually or in combination, was verified and demonstrates how its use led to positive outcomes. However, this work is always subject to further evaluations, yet, deeper investigations are at hand. The authors provide guidance for others who need to improve their Arabic classifiers, especially that the tools selected are available to the research community.

Finally, the results reported here open up avenues for further research in order to advance the state of Arabic text classification. The next objective of this study will be investigating other ways to improve the classification task focusing on other phases, namely feature extraction, feature selection, and the classification algorithms. Besides, building well-designed language resources for both single and multi-category Arabic text classification is always recommended.

REFERENCES

1. I. Guellil, H. Saâdane, F. Azouaou, B. Gueni, and D. Nouvel, **Arabic natural language processing: An overview**, *Journal of King Saud University - Computer and Information Sciences*, Feb. 2019.
2. I. Zeroual and A. Lakhouaja, **Arabic Corpus Linguistics: Major Progress, but Still a Long Way to**

Go, in *Intelligent Natural Language Processing: Trends and Applications*, Springer, Cham, 2018, pp. 613–636.
3. O. Obeid, N. Zalmout, S. Khalifa, D. Taji, M. Oudah, B. Alhafni, G. Inoue, F. Eryani, A. Erdmann, and N. Habash, **CAMeL tools: An open source python toolkit for Arabic natural language processing**, in *Proceedings of the 12th language resources and evaluation conference*, 2020, pp. 7022–7032.
4. M. Sayed, R. K. Salem, and A. E. Khder, **A survey of Arabic text classification approaches**, *International Journal of Computer Applications in Technology*, Vol. 59, no. 3, pp. 236–251, Jan. 2019.
5. I. Zeroual, D. Goldhahn, T. Eckart, and A. Lakhouaja, **OSIAN: Open Source International Arabic News Corpus - Preparation and Integration into the CLARIN-infrastructure**, in *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, Florence, Italy, 2019, pp. 175–182.
6. F. S. Al-Anzi and D. AbuZeina, **Stemming impact on Arabic text categorization performance: A survey**, in *2015 5th International Conference on Information Communication Technology and Accessibility (ICTA)*, 2015, pp. 1–7.
7. L. S. Larkey, L. Ballesteros, and M. E. Connell, **Light stemming for Arabic information retrieval**, in *Arabic computational morphology*, Springer, 2007, pp. 221–243.
8. S. Khoja and R. Garside, **Stemming arabic text**, *Lancaster, UK, Computing Department, Lancaster University*, 1999.
9. A. Wahbeh, M. Al-Kabi, Q. Al-Radaideh, E. Al-Shawakfa, and I. Alsmadi, **The effect of stemming on arabic text classification: an empirical study**, *International Journal of Information Retrieval Research (IJIRR)*, Vol. 1, no. 3, pp. 54–70, 2011.
10. D. Namly, K. Bouzoubaa, A. El Jihad, and S. L. Aouragh, **Improving Arabic Lemmatization Through a Lemmas Database and a Machine-Learning Technique**, in *Recent Advances in NLP: The Case of Arabic Language*, Springer, 2020, pp. 81–100.
11. A. Chouigui, O. B. Khiroun, and B. Elayeb, **ANT corpus: an Arabic news text collection for textual classification**, in *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, 2017, pp. 135–142.
12. D. Namly, K. Bouzoubaa, R. Tajmout, and A. Laadimi, **On Arabic Stop-Words: A Comprehensive List and a Dedicated Morphological Analyzer**, in *Arabic Language Processing: From Theory to Practice*, Cham, 2019, pp. 149–163.
13. A. Alajmi, E. M. Saad, and R. R. Darwish, **Toward an ARABIC stop-words list generation**, *International Journal of Computer Applications*, Vol. 46, no. 8, pp. 8–13, 2012.
14. I. A. El-Khair, **Effects of stop words elimination for Arabic information retrieval: a comparative study**, *arXiv preprint arXiv:1702.01925*, 2017.
15. R. Z. Al-Abdallah and A. T. Al-Taani, **Arabic single-document text summarization using particle swarm optimization algorithm**, *Procedia Computer Science*, Vol. 117, pp. 30–37, 2017.
16. K. K. Arora and S. S. Agrawal, **Pre-Processing of English-Hindi Corpus for Statistical Machine Translation**, *Computación y Sistemas*, Vol. 21, no. 4, pp. 725–737, 2017.
17. Z. Jianqiang and G. Xiaolin, **Comparison Research on Text Pre-processing Methods on Twitter Sentiment Analysis**, *IEEE Access*, Vol. 5, pp. 2870–2879, 2017.
18. I. Zeroual, M. Boudchiche, A. Mazroui, and A. Lakhouaja, **Developing and Performance Evaluation of a New Arabic Heavy/Light Stemmer**, in *Proceedings of the 2Nd International Conference on Big Data, Cloud and Applications*, Tetouan, Morocco, 2017, p. 17:1-17:6.
19. O. A. Ghanem and W. M. Ashour, **Stemming Effectiveness in Clustering of Arabic Documents**, *International Journal of Computer Applications*, Vol. 49, no. 5, pp. 1–6, Jul. 2012.
20. S. Bahassine, A. Madani, and M. Kissi, **Arabic text classification using new stemmer for feature selection and decision trees**, *Journal of Engineering Science and Technology*, Vol. 12, no. 6, pp. 1475–1487, 2017.
21. M. Naili, A. H. Chaibi, and H. H. B. Ghezala, **Comparative study of Arabic stemming algorithms for topic identification**, *Procedia Computer Science*, Vol. 159, pp. 794–802, 2019.
22. A. Abdelali, K. Darwish, N. Durrani, and H. Mubarak, **Farasa: A fast and furious segmenter for arabic**, in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Demonstrations*, 2016, pp. 11–16.
23. M. Boudchiche, A. Mazroui, M. O. A. O. Bebah, A. Lakhouaja, and A. Boudlal, **AlKhalil Morpho Sys 2: A robust Arabic morpho-syntactic analyzer**, *Journal of King Saud University-Computer and Information Sciences*, Vol. 29, no. 2, pp. 141–146, 2017.
24. Y. A. Alhaj, J. Xiang, D. Zhao, M. A. Al-Qaness, M. Abd Elaziz, and A. Dahou, **A study of the effects of stemming strategies on arabic document classification**, *IEEE Access*, Vol. 7, pp. 32664–32671, 2019.
25. K. Taghva, R. Elkhoury, and J. Coombs, **Arabic stemming without a root dictionary**, in *null*, 2005, pp. 152–157.
26. K. Abainia, S. Ouamour, and H. Sayoud, **A novel robust Arabic light stemmer**, *Journal of Experimental & Theoretical Artificial Intelligence*, pp. 1–17, 2016.
27. K. Abainia and H. Rebbani, **Comparing the Effectiveness of the Improved ARLSTem Algorithm with Existing Arabic Light Stemmers**, in *2019 International Conference on Theoretical and Applicative Aspects of Computer Science (ICTAACS)*, 2019, Vol. 1, pp. 1–8.

28. H. Soori, J. Platoš, and V. Snášel, **Simple stemming rules for Arabic language**, in *Proceedings of the Third International Conference on Intelligent Human Computer Interaction (IHCI 2011), Prague, Czech Republic, August, 2011*, 2013, pp. 99–108.
29. I. Zeroual and A. Lakhouaja, **Arabic information retrieval: Stemming or lemmatization?**, in *2017 Intelligent Systems and Computer Vision (ISCV)*, Fez, Morocco, 2017, pp. 1–6.
30. T. El-Shishtawy and F. El-Ghannam, **A Lemma Based Evaluator for Semitic Language Text Summarization Systems**, *arXiv preprint arXiv:1403.5596*, 2014.
31. F. K. Hammouda and A. A. Almarimi, **Heuristic Lemmatization for Arabic Texts Indexation and Classification**, *Journal of Computer Science*, Vol. 6, no. 6, pp. 660–665, Jun. 2010.
32. A. Pasha, M. Al-Badrashiny, M. T. Diab, A. El Kholy, R. Eskander, N. Habash, M. Pooleery, O. Rambow, and R. Roth, **MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic.**, in *LREC*, 2014, Vol. 14, pp. 1094–1101.
33. S. R. Garner, **Weka: The waikato environment for knowledge analysis**, in *Proceedings of the New Zealand computer science research students conference*, 1995, Vol. 1995, pp. 57–64.
34. L. A. Qadi, H. E. Rifai, S. Obaid, and A. Elnagar, **Arabic Text Classification of News Articles Using Classical Supervised Classifiers**, in *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*, 2019, pp. 1–6.
35. A. Alahmadi, A. Joorabchi, and A. E. Mahdi, **Combining Words and Concepts for Automatic Arabic Text Classification**, in *Arabic Language Processing: From Theory to Practice*, Cham, 2018, pp. 105–119.
36. P. Kumbhar and M. Mali, **A survey on feature selection techniques and classification algorithms for efficient text classification**, *International Journal of Science and Research*, Vol. 5, no. 5, p. 9, 2016.
37. X. Deng, Y. Li, J. Weng, and J. Zhang, **Feature selection for text classification: A review**, *Multimed Tools Appl*, Vol. 78, no. 3, pp. 3797–3816, Feb. 2019.
38. A. S. Ghareb, A. A. Bakara, Q. A. Al-Radaideh, and A. R. Hamdan, **Enhanced filter feature selection methods for Arabic text categorization**, *International Journal of Information Retrieval Research (IJIRR)*, Vol. 8, no. 2, pp. 1–24, 2018.
39. H. Chantar, M. Mafarja, H. Alsawalqah, A. A. Heidari, I. Aljarah, and H. Faris, **Feature selection using binary grey wolf optimizer with elite-based crossover for Arabic text classification**, *Neural Computing and Applications*, Vol. 32, no. 16, pp. 12201–12220, 2020.
40. N. S. M. Nafis and S. Awang, **The Evaluation of Accuracy Performance in an Enhanced Embedded Feature Selection for Unstructured Text Classification**, *Iraqi Journal of Science*, pp. 3397–3407, 2020.
41. M. Hijazi, A. Zeki, and A. Ismail, **Arabic Text Classification Using Hybrid Feature Selection Method Using Chi-Square Binary Artificial Bee Colony Algorithm**, *Computer Science*, Vol. 16, no. 1, pp. 213–228, 2021.
42. H. Tang, L. Zhou, X. Chengjie, and Q. Zhu, **A Method of Text Dimension Reduction Based on CHI and TF-IDF**, in *2015 4th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering*, 2015.
43. T. Joachims, **Text categorization with support vector machines: Learning with many relevant features**, in *European conference on machine learning*, 1998, pp. 137–142.