applied sciences

MDPI

Article

Advancing Arabic Word Embeddings: A Multi-Corpora Approach with Optimized Hyperparameters and Custom Evaluation

Azzah Allahim [1,2,*,†] and Asma Cherif [1,3,†]

[1] IT Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia; acherif@kau.edu.sa
[2] College of Computer and Information Sciences, Jouf University, Sakaka 72388, Saudi Arabia
[3] The Center of Excellence in Smart Environment Research, King Abdulaziz University, Jeddah 21589, Saudi Arabia
* Correspondence: azzah.allahim@gmail.com
† These authors contributed equally to this work.

Abstract: The expanding Arabic user base presents a unique opportunity for researchers to tap into vast online Arabic resources. However, the lack of reliable Arabic word embedding models and the limited availability of Arabic corpora poses significant challenges. This paper addresses these gaps by developing and evaluating Arabic word embedding models trained on diverse Arabic corpora, investigating how varying hyperparameter values impact model performance across different NLP tasks. To train our models, we collected data from three distinct sources: Wikipedia, newspapers, and 32 Arabic books, each selected to capture specific linguistic and contextual features of Arabic. By using advanced techniques such as Word2Vec and FastText, we experimented with different hyperparameter configurations, such as vector size, window size, and training algorithms (CBOW and skip-gram), to analyze their impact on model quality. Our models were evaluated using a range of NLP tasks, including sentiment analysis, similarity tests, and an adapted analogy test designed specifically for Arabic. The findings revealed that both the corpus size and hyperparameter settings had notable effects on performance. For instance, in the analogy test, a larger vocabulary size significantly improved outcomes, with the FastText skip-gram models excelling in accurately solving analogy questions. For sentiment analysis, vocabulary size was critical, while in similarity scoring, the FastText models achieved the highest scores, particularly with smaller window and vector sizes. Overall, our models demonstrated strong performance, achieving 99% and 90% accuracies in sentiment analysis and the analogy test, respectively, along with a similarity score of 8 out of 10. These results underscore the value of our models as a robust tool for Arabic NLP research, addressing a pressing need for high-quality Arabic word embeddings.

Keywords: word embedding; Word2vec; FastText; Arabic embedding; Arabic corpus

Citation: Allahim, A.; Cherif, A. Advancing Arabic Word Embeddings: A Multi-Corpora Approach with Optimized Hyperparameters and Custom Evaluation. *Appl. Sci.* **2024**, *14*, 11104. https://doi.org/10.3390/app142311104

Academic Editor: Jose María Alvarez Rodríguez

Received: 29 September 2024
Revised: 17 November 2024
Accepted: 20 November 2024
Published: 28 November 2024

1. Introduction

Arabic has linguistic complexities such as morphological richness, complex syntax, and multiple dialects spread over many geolinguistic environments, which affect Arabic natural language processing (NLP) task performance compared to several other languages. Arabic word embedding plays a crucial role in Arabic NLP research. Word embedding is mainly a type of distributed word representation that captures both the semantic and syntactic information of words. It is derived from a large unlabeled corpus of text data [1,2]. Indeed, word embedding is a valuable technique that provides numerical representations of words, enabling meaningful comparisons. Therefore, word embedding models are trained on large text corpus using algorithms such as Word2Vec, GloVe, or FastText. These pretrained models provide a robust method for representing Arabic words in a high-dimensional

vector space. By employing statistical techniques to analyze frequently occurring n-grams from diverse Arabic content domains, such as Tweets and Arabic Wikipedia articles, these models are generated. This vector representation enables researchers to enhance the effectiveness and accuracy of various NLP tasks. Moreover, these models can be easily downloaded and integrated into numerous NLP applications.

Mikolov et al. [3] have made significant contributions to this field with their widely used Word2Vec model. This model efficiently learns vector representations of words from large text datasets, capturing semantic relationships and enhancing various NLP tasks such as sentiment analysis, machine translation, and information retrieval. However, their work is developed using English corpora and is primarily useful for English NLP tasks [4]. Furthermore, Facebook has published an enhanced version of Word2Vec called FastText [5]. FastText incorporates the use of skip-gram models, which involve utilizing n-grams for each word. This approach enables faster training on large corpora and also allows the model to handle new words that were not present in the training data. The FastText model has gained popularity due to its efficiency and ability to capture the nuances of language [6].

Pretrained models, which are widely available to the public and trained on extensive datasets, have become a standard resource for various NLP applications. However, it is important to note that these models are primarily focused on the English language [7]. Indeed, these models greatly facilitate the efficient calculation of word similarities by utilizing low-dimensional matrix operations. This capability proves to be valuable in addressing the complexities of Arabic language features and mitigating issues related to semantic ambiguity. By representing Arabic words in a high-dimensional vector space, these models can capture the subtle nuances and context-specific meanings of Arabic words, enabling more accurate and effective analysis in Arabic natural language processing (NLP) tasks [8].

Despite the growing popularity of word embedding techniques, the development of Arabic word embedding is still in its early stages. The main hurdle lies in the scarcity of Arabic language resources. The creation of a robust Arabic corpus is challenging due to the limited availability of reliable sources. Evaluating Arabic word embedding models requires diverse datasets, incorporating analogy tests, sentiment-annotated data, and word pairs with synonymous meanings. These datasets are crucial for assessing a model's ability to accurately capture semantic relationships in the Arabic language.

Only a few works have been published on Arabic word embedding models. For instance, in [9], the authors developed Arabic models with various hyperparameter values. However, these models are not readily available for use. Additionally, the researchers collected their corpora from online resources and utilized a translated version of the Google Analogy Test dataset. Despite its popularity, the structure of the Google Analogy Test dataset in terms of relations has been questioned. Many works have used a translated version of the Google Analogy Test for non-English word embedding models; however, its efficiency in evaluating non-English word embedding models has been criticized [10]. For morphologically rich languages such as Arabic and German, using the Google Analogy Test may not be effective since it does not provide a deep representation of these languages from the outset. One of the major issues with using the Google Analogy Test is that it contains a (city-in-state) test section, which is irrelevant to some languages, such as Arabic, due to cultural differences. Additionally, some vocabulary used in the test does not have exact translations, leading the model to fail in producing the desired results. For example, in the (family) test section, the word (stepbrother) cannot be translated as a single word in Arabic.

Another published Arabic model is Aravec [11], which is available for use. However, it has limitations in capturing certain Arabic words. Additionally, the authors did not publish the details of the corpora used or the testing datasets employed.

From previous efforts in Arabic word embedding, we can conclude that these models face several limitations, including limited availability, reliance on unsuitable evaluation tests, such as the Google Analogy Test, vocabulary loss due to translation challenges, and a lack of transparency in corpus and test dataset details. These limitations high-

light the need for more comprehensive and openly available Arabic word embedding models and resources.

Therefore, to address this gap, this paper proposes the development of a comprehensive set of high-quality Arabic word embedding models that can be effectively utilized for various natural language processing (NLP) tasks, including sentiment analysis and similarity testing. The contributions of this paper are as follows:

- The creation of comprehensive Arabic embedding models: We developed 108 Arabic word embedding models using advanced techniques such as Word2Vec and FastText, optimizing the hyperparameters to produce high-quality embeddings specifically suited for the unique linguistic structures of Arabic.
- Rich language representation: By training the models on three diverse Arabic corpora—news articles, books, and Wikipedia—we capture the linguistic variety and morphological richness of Arabic, supporting robust, context-aware embeddings.
- The development of Arabic-specific evaluation metrics: We adapted an Arabic benchmark from Google's Analogy Test to accurately assess semantic relationships within Arabic, ensuring the embeddings reflect true language understanding.

In addition to the aforementioned contributions, this paper provides open access to the developed models, corpora, and testing datasets (Our work is available at https://github.com/AzzahAllahim/ArabicWordEmbedding (accessed on 28 September 2024)). By doing so, we strive to contribute valuable tools that can enhance the Arabic NLP domain.

The remainder of this paper is organized as follows. Section 2 provides background information about Word embeddings. Related works are summarized in Section 3. Section 4 elaborates on the proposed Arabic word embedding model-building process. This is followed by their evaluation in Section 5 and the discussion of the results in Section 6. Finally, we conclude in Section 7.

2. Research Background

Word embedding is a technique that assigns numerical values to words, enabling them to be compared and analyzed quantitatively. It involves converting words, represented as strings, into vectors in a vector space based on selected features or dimensions. Typically, word embedding represents each word as a one-hot vector. These vectors are then mapped in a continuous space, placing semantically similar words closer to each other.

Word embedding has proven to be highly valuable in various NLP applications. It provides an efficient means for machines to understand the meaning of words and successfully perform NLP tasks. Notably, there are several state-of-the-art approaches to word embedding, such as Word2vec, GloVe, and FastText. These methods have significantly contributed to advancing the field of natural language processing.

Word2vec. This was introduced by Mikolov et al. [3]. The main idea behind Word2Vec is to pretrain a single projection matrix that incorporates the embedding dimensions and the vocabulary. The embedding is structured by maximizing the likelihood of word prediction given their context. Word2Vec can be constructed using either the skip-gram or continuous bag of words (CBOW) methods [12]. In skip-gram, the model predicts the surrounding context words based on the input word. On the other hand, CBOW uses the words surrounding the target word to predict it (see Figure 1). Both methods employ the Softmax function to assign probabilities to each word in the vocabulary as the output layer. Hierarchical Softmax is employed to reduce the search time for output nodes. This is achieved by representing the vocabulary as a Huffman binary tree [13], enabling faster search by focusing on only half of the vocabulary.

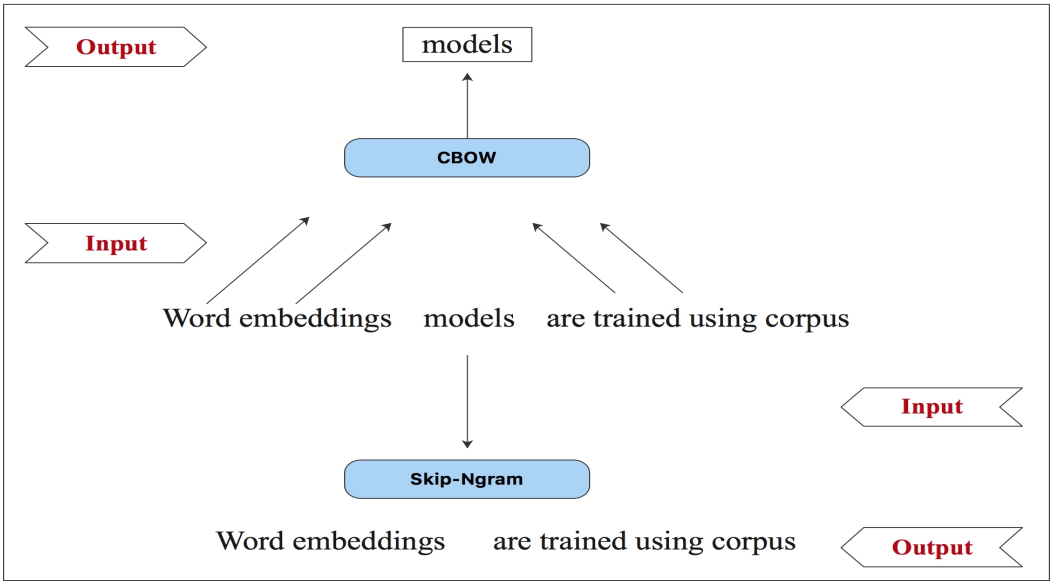


Figure 1. Illustration of the skip-gram and continuous bag-of-word (CBOW) models.

GloVe. This was introduced by Pennington et al. [14], who proposed the idea that the ratio of word–word co-occurrence probabilities can indicate the meaning of words. Word2Vec utilizes this concept and is trained using a global corpus of word occurrences to generate linear directions of meanings [9]. The primary method employed by the GloVe approach is to utilize a textual corpus to capture word occurrences. This is achieved by constructing a co-occurrence matrix based on the co-occurrence frequency of each word with other words. The approach takes into account a fixed window size context to determine the co-occurrence scores. However, the resulting matrix is symmetric, which provides less useful information about the relatedness of the words. To overcome this limitation, GloVe relies on calculating co-occurrence ratios between two words in a context. Additionally, the matrix is factorized to generate vectors, which are then used for word embedding.

FastText. FastText is a word embedding model developed by the Facebook research team. Unlike previous models, FastText considers words as chunks of characters, allowing them to be formed by character n-grams. An n-gram is represented by a vector, and the word vector is the sum of the n-gram vectors. This unique feature enables FastText to effectively represent rare words by creating vectors for words that may not exist in the training corpus [9]. Additionally, words composed of common roots can be used to learn new vocabulary that infrequently appears in the corpus. This capability is particularly advantageous when dealing with out-of-vocabulary words, as FastText can still provide meaningful representations based on their character-level composition. Furthermore, FastText can overcome the limitation in representing slang and misspelled words [15].

There are two approaches to using word embedding models. The first approach is to utilize a pretrained model that satisfies the language requirements. The second approach involves training one of the aforementioned state-of-the-art models. In the first approach, a suitable pretrained model is selected based on its compatibility with the desired language. The model's hyperparameters are then adjusted to meet specific goals. In the second approach, a custom model is trained using four main steps: collecting a corpus for the desired language, performing data preprocessing on the corpus, setting up the model's hyperparameters, and evaluating the training.

The corpus can be collected from various sources, such as Wikipedia or social media platforms. Hyperparameters, such as the contextual window length and vector size (number of dimensions), are typically tuned during training. To evaluate the training quality, two types of evaluations can be conducted: intrinsic and extrinsic.

Intrinsic evaluations involve testing the model using general-purpose tasks such as word analogy and concept categorization. Extrinsic evaluations, on the other hand, test the model on specific NLP tasks to assess its performance in real-world applications.

3. Related Work

Arabic, a widely spoken language with 422 million speakers across 27 countries [16], faces a significant shortage of resources when it comes to building Arabic word embedding models. This scarcity is evident in the limited number of attempts made in this area. Nonetheless, there have been commendable efforts in this regard. Two notable works [9,11] have made substantial contributions to the field of Arabic word embeddings. Despite the challenges, these works have made significant strides in enhancing our understanding of Arabic language representation in computational models.

Soliman [11] introduced "AraVec", an open source project for word embeddings. AraVec aims to provide the Arabic NLP research community with free and powerful word embedding models. The project utilized Twitter, the World Wide Web, and Wikipedia as training resources and developed six different models using CBOW and skip-gram structures. Their quantitative evaluation revealed that the skip-gram model with the Twitter dataset produced the most promising results.

Additionally, Chaimae et al. [9] conducted a comparative study of word embeddings, creating 180 different models using Word2vec, GloVe, and FastText. Their dataset sources included Wikipedia and various social media platforms. In their evaluations, intrinsic and extrinsic assessments were employed. Their findings indicated that Word2vec models performed well in applications with a semantic nature, such as sentiment analysis, while GloVe outperformed Word2vec in applications with syntactic and contextual characteristics.

Table 1 provides a brief comparison between our work and Refs. [9,11] regarding various aspects such as vector size, window size, Arabic analogy consideration, the availability of models, the availability of corpora, and the usage of a Books corpus.

Table 1. A brief comparison between our paper and other Arabic word embeddings.

Ref.	Multiple Vector Size	Multiple Window Size	Arabic Analogy	Models Availability	Corpus Availability	Books Corpus
AraVec [11]				✓		
Azroumahli et al. [9]	✓	✓				
Our paper	✓	✓	✓	✓	✓	✓

As illustrated in Table 1, the work in [11] did not consider various hyperparameter values, a crucial consideration given that the optimal values for Arabic models have not yet been discovered. Furthermore, it did not include analogy testing in the model evaluation. In contrast, while the work in [9] used analogy testing, they utilized a translated version of the Google Analogy Test, which does not provide a fair evaluation. Criticisms have been raised regarding the efficiency of the Google Analogy Test for evaluating non-English word embedding models, particularly for morphologically rich languages such as Arabic and German [10]. The test's depth representation of these languages is insufficient, and it contains a (city-in-state) test section that is culturally irrelevant to languages such as Arabic. Additionally, some vocabulary used in the test may not have exact translations, leading to model failure in producing target results. The availability of both Aravec [11] and other models [9] is also a concern, as they are not publicly accessible.

To overcome the previously mentioned challenge, we propose building Arabic word embedding models by utilizing existing Arabic books to enrich the corpus. Furthermore, we will experiment with various hyperparameter combinations to construct the models and assess the impact of each combination.

For our evaluation, we employed an Arabic analogy test derived from [10] and a modified translated version of the Google Analogy Test. By making all the models

and corpora in our work available, we aim to contribute to the enrichment of Arabic NLP and address the current scarcity of resources in the field.

4. Building Arabic Word Embeddings

We constructed our methodology to build Arabic word embeddings based on two main components: the corpus and the training method. Therefore, as illustrated in Figure 2, our methodology relies on three main steps: corpus collection, corpus preprocessing, and model training.

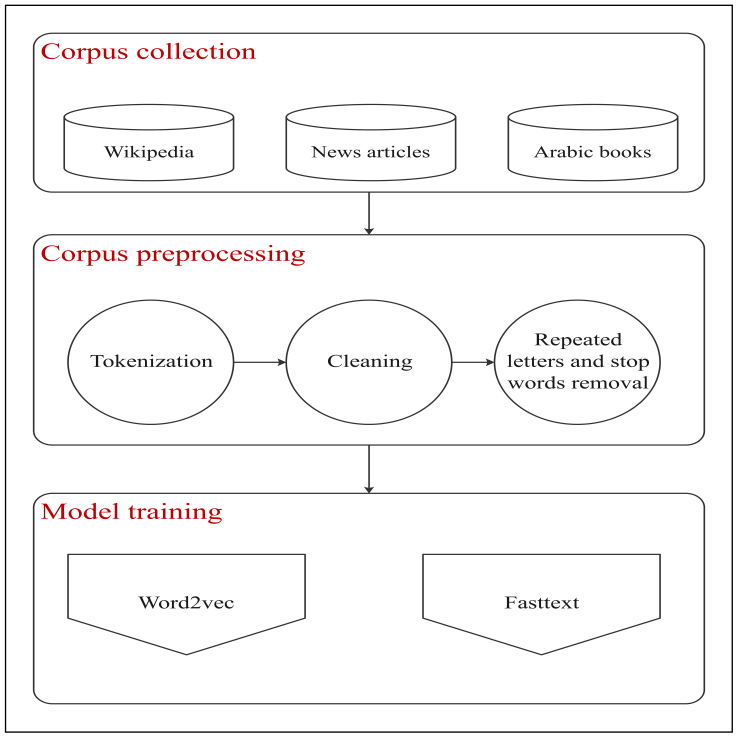


Figure 2. Building Arabic Word Embeddings Methodology

Three distinct corpora were employed for training. The primary factor in effectively training a specific word embedding model is the selection of a comprehensive corpus, offering a diverse range of vocabularies across various topics. Constructing a corpus involves two key steps: selecting an appropriate source for corpus collection and preprocessing its contents. Then, to train our models, we utilized Word2Vec and FastText with CBOW and skip-gram structures.

In the following section, we provide a detailed explanation of the steps involved in constructing our corpora, which encompasses corpus collection and preprocessing. Subsequently, we describe the training process of various models with diverse hyperparameter values using each corpus.

4.1. Corpus Collection

For our research, we gathered three distinct corpora from various sources: the Watan corpus, the Books corpus, and the Wiki corpus. Our objective in creating these corpora was to achieve our specific research goals, which involve ensuring that the word vectors capture a diverse range of nuances and variations within the Arabic language.

To create a morphologically rich corpus, we carefully selected 32 Arabic books that extensively represent Arabic words in different forms. Additionally, each word in the corpus is accompanied by an explanation, enhancing the semantic depth of the word vectors, courtesy of the window parameter of the models. For the moderately topical corpus, we curated news articles known for their diverse subject matters. Lastly, we opted

for Wikipedia as the source of the large corpus due to its vast collection of articles. Here, we present a comprehensive overview of each corpus:

- **Book corpus**: Classical Arabic is one of the Arabic varieties that is commonly used in religious contexts and is distinguished by its unique and formal structure compared to modern and dialectical Arabic. Regrettably, classical Arabic, particularly ancient Arabic, has not received significant attention in the field of word embeddings. To address this gap, we constructed our first corpus by utilizing 32 well-known Arabic books. The digital versions of these books were obtained from the Almaktabah Alshamelah database (https://shamela.ws/ (accessed on 28 September 2024)). This database offers a wealth of Arabic words along with detailed descriptions. Table 2 shows the topics' distribution in the Books corpus. Notably, one of these books, Lesan Al-arab, is considered to be the comprehensive dictionary of all Arabic dictionaries [17]. Given that these books provide various forms and descriptions for numerous Arabic words, they contribute to the morphological richness of the embedding model. Through the use of this corpus, we aim to analyze the efficacy of employing Arabic books for word embedding models and to assess their ability to capture rare and ancient Arabic words. This corpus contains approximately 11 million words.
- **Watan corpus**: To build a topically rich medium corpus, we used the Arabic corpus Watan-2004 [18]. This corpus contains 20,000 Arabic news articles covering a wide range of topics such as culture, religion, economy, local news, international news, and sports, all provided in HTML format; additional information about the corpus can be found in [18]. Table 3 shows the topics' distribution in the Watan corpus. This corpus contains approximately 6 million words.

Table 2. Categories distribution in the Books corpus.

Category	Number of Books
Dictionaries	8
Linguistics	9
Gramertic	15

Table 3. Topics distribution in the Watan corpus.

Topic	Number of Articles
Culture	2782
Religion	3860
Economy	3468
Local news	3596
Internation news	2035
Sports	4550

- **Wiki corpus**: Wikipedia (https://www.wikipedia.org/ (accessed on 28 September 2024)) provides a plethora of articles on various topics in multiple languages. Therefore, we have selected Wikipedia as the source of the large corpus. We collected recent articles from 2023 using the Wikiextractor tool from [19] to gather articles from Wikimedia, a website that archives articles in different languages (https://dumps.wikimedia.org/ (accessed on 28 September 2024)). The collected files comprise recent Arabic articles, totaling around 2 million articles across different topics. This corpus contains approximately 111 million words.

Table 4 summarizes the sources, the total word size, and the total unique vocabulary for each corpus. As shown in the table, in the Book corpus, we were able to obtain 40.69% of the Wikipedia vocabulary size by using only 9.91% of the total words compared with Wikipedia. This indicates that Arabic books can be utilized to create valuable corpora, which could help narrow the resource gap in Arabic NLP. In the following, we will delve

into the preprocessing steps for each corpus, which involve cleaning and preparing the contents to be more useful.

Table 4. Summery of the corpora.

Corpus	Source	Number of Words	Number of Vocabulary
Watan	Watan newspaper	6 M	35,211
Book	32 Arabic books	11 M	181,432
Wiki	Wikipedia	111 M	445,977

4.2. Corpus Preprocessing

Preprocessing is a critical step in preparing a corpus for training. Feeding a model with a corpus containing redundant vocabulary can result in unwanted output. To prevent the development of inadequate models, we implement several preprocessing steps on our corpora. Figure 3 illustrates the main steps of the corpus preprocessing. It includes tokenization, cleaning, normalization, repeated letters removal, and stop words removal.

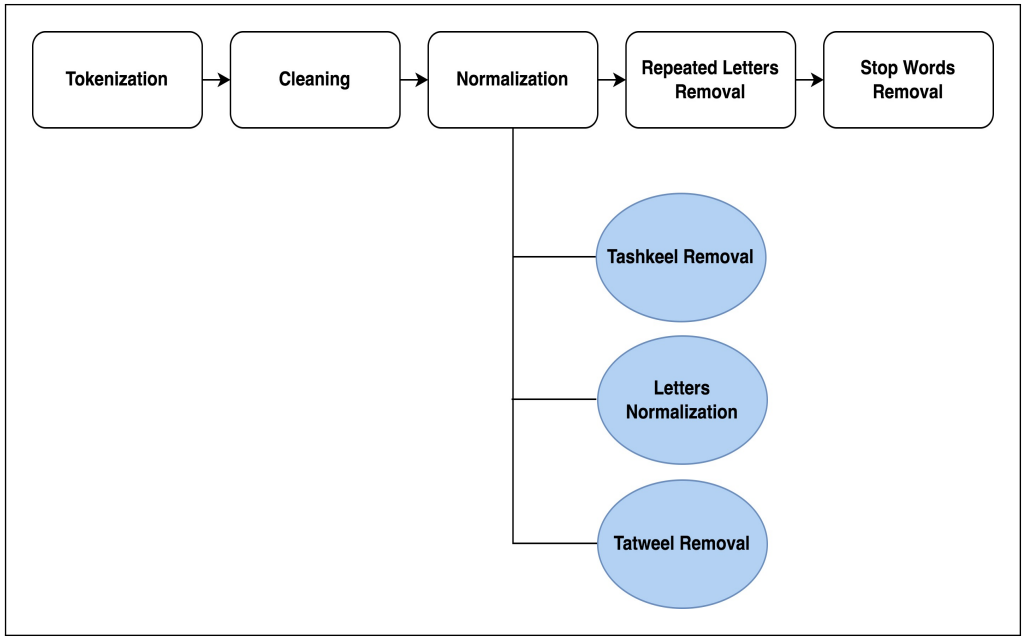


Figure 3. Illustration of the corpus preprocessing steps.

- **Tokenization**: Since the text is considered as a single input block for the preprocessing program, we utilized tokenization as the first step. Tokenization breaks the text into individual tokens (words), allowing for easier processing. To accomplish this, we employed the (tokenize) function, which is a part of the pyarabic (https://pypi.org/project/PyArabic/ (accessed on 28 September 2024)) library.
- **Cleaning**: Then, we cleaned the text to remove any unwanted data. This includes removing digits, non-Arabic text, punctuation marks, emojis, and symbols.
- **Normalization**: Arabic is a highly complex language due to its richness, which includes representing the same word in various formats. To enhance the robustness of our models, we implemented several normalization steps to prevent word redundancy.
  - **Tashkeel Removal**: One of the unique features of Arabic is its use of specific marks for each letter in words. These marks represent the required sound for each letter in the word. Their existence affects the representation of the word, as changing one mark can lead the model to perceive the word as a new one. From the model's perspective, every unique input is treated as individual input. To prevent the model from establishing different vectors for each marked version

of the word, we removed any tashkeel marks. To perform this, we used the (strip_tashkeel) function from the (pyarabic) library.
  - **Letters Normalization**: We conducted letter normalization to standardize the format of certain Arabic letters. This step is crucial in ensuring that the model treats each letter consistently, preventing redundancy and improving overall performance.
  - **Tatweel Removal**: Some Arabic writers use (Tatweel) to add a touch of elegance to the text by incorporating extra dashes within the words. We removed it from the text by utilizing the (strip_tatweel) function from the (pyarabic) library.
- **Repeated Letters Removal**: One of the challenges that affects our training is the presence of unintentionally repeated letters within the same word. To address this issue, we implemented a step to remove repeated letters within the same word.
- **Stop Words Removal**: In the context of the Arabic language, stop words are words that are frequently used but do not carry significant meaning. By removing these stop words, we can reduce the size of the corpus and focus on more meaningful words. We started by adopting the Arabic stop words list provided by NLTK (https://www.nltk.org/ (accessed on 28 September 2024)), which contains 754 words. However, we noticed that stop words in Arabic can appear in various forms, such as different tenses, formats, or with the conjunction letter attached directly to them. To address this, we expanded the NLTK Arabic stop words list by adding a conjunction letter to each word and applied normalization to words that can come in different formats. Additionally, we added present and future verb formats to the stop words list. The resulting stop words list contains around 936 words.

4.3. Model Training

To train our Arabic models, we employed two algorithms: Wor2Vec and FastText. For the Wor2Vec algorithm, we utilized the Gensim library, while the FastText library was used for the FastText models. Mikolov et al. [3] emphasized that the most critical hyperparameters for word embedding models are architecture, dimension size, and context window size. Consequently, we constructed various models based on different combinations of these hyperparameters.

The architecture hyperparameter pertains to the utilized training algorithm, whether it is a skip-gram or CBOW. Additionally, the models input the words as numerical values, known as vectors. The dimension size refers to the vector size for each word, determining the number of features that represent the word. It can range from 100 to 1000. Unfortunately, there is no research available that provides the optimal hyperparameter values for Arabic. To avoid overfitting or underfitting problems for our models, we chose 200, 300, and 400 as the vector sizes.

Lastly, for a morphologically rich language such as Arabic, the context window size plays a significant role in training the models. Essentially, the context window size refers to the number of words surrounding a specific word (target word). It determines the attention range for the model regarding the specific word. This means that the surrounding words are part of the word prediction. Given Arabic's rich syntax and semantic properties, we selected 5, 7, and 10 as window sizes.

By taking into account the three different corpora and the previously chosen hyperparameter values, the total number of possible combinations is 108. Consequently, we trained 108 distinct models. Among these models, those trained with the Wikipedia corpus required the longest time, with each model taking 8 h to complete. In contrast, the models trained with the Books corpus took 25 min per model, and the Watan corpus models took approximately 10 min per model.

5. Model Evaluation

To assess the quality of the training, two distinct evaluations can be conducted: intrinsic and extrinsic evaluations [20]. In intrinsic evaluation, the model is tested using general-purpose tasks such as word analogy tasks and concept categorization. Conversely,

extrinsic evaluation involves testing the model using specific-purpose tasks such as NLP tasks. For our study, we conducted the analogy test as an intrinsic evaluation. For the extrinsic evaluation, we conducted sentiment clustering and a semanticity test.

5.1. Analogy Test

The primary objective of the analogy test is to determine the term D based on the given terms A, B, and C. Essentially, given that A:B are related by a specific type of relationship, C would be associated with D using the same relationship. Therefore, the test is structured as follows: if (A,B), then (C-?). The model should respond to this question by predicting the term D. This equation is primarily employed to ascertain the answer:

$$A - B + C = D \tag{1}$$

One of the famous examples of an analogy test question is: "king – man + woman = queen".

To conduct this test, a predefined analogy is utilized, containing pairs of items that are related in a specific manner, such as country–capital, country–currency, and singular–plural. One widely used analogy benchmark is the Google Analogy Test developed by Mikolov et al. [3]. It encompasses 14 types of relations (nine morphological and five semantic) and serves as a popular benchmark for evaluating word embedding models. Figure 4 depicts a subset of the pairs from the Google Analogy Test.

```
: capital-common-countries
Athens Greece Baghdad Iraq
Athens Greece Bangkok Thailand
Athens Greece Beijing China
Athens Greece Berlin Germany
Athens Greece Bern Switzerland
Athens Greece Cairo Egypt
Athens Greece Canberra Australia
Athens Greece Hanoi Vietnam
Athens Greece Havana Cuba
Athens Greece Helsinki Finland
Athens Greece Islamabad Pakistan
Athens Greece Kabul Afghanistan
Athens Greece London England
Athens Greece Madrid Spain
Athens Greece Moscow Russia
Athens Greece Oslo Norway
Athens Greece Ottawa Canada
Athens Greece Paris France
```

Figure 4. A subset of the Google Analogy Test.

Despite its popularity, there have been doubts about the structure of the Google Analogy Test in terms of relations. The Google Analogy Test contains limited relations. When assessing non-English word embedding models, many studies have utilized a translated version of the Google Analogy Test. However, its effectiveness in evaluating non-English word embedding models has been criticized [10]. For languages with rich morphology, such as Arabic and German, using the Google Analogy Test could result in shortcomings, as it does not initially offer a comprehensive representation of these languages.

One major issue with the Google Analogy Test is the inclusion of the "city-in-state" test section, which is irrelevant to certain languages, such as Arabic, due to cultural differences. Furthermore, some vocabulary used in the test lacks an exact translation, leading to challenges for the model in producing the intended results. For example, in the "family" test section, the words "nephew" and "niece" cannot be accurately translated as single words in Arabic. It is noteworthy that limited efforts have been made to provide a suitable version of the Analogy Test dataset for Arabic. Most of these efforts have involved translating the Google Analogy Test, as observed in [21,22], wherein it was argued that translating an English benchmark is challenging because it could result in translation drift. Additionally, the Google Analogy Test was specifically designed for English, and some

of its pairs may not be applicable to Arabic. Moreover, Yagi et al. [10] asserted that the existing analogies are insufficient for other languages that have unique characteristics. For instance, the Arabic language is root-based, morphologically rich, and has complex syntax.

These limitations have led some researchers to adapt and expand the Google Analogy Test using further adjustments. For instance, Gladkova et al. [23] contended that the Google Analogy Test has limited relations. Consequently, they created the Bigger Analogy Test Set, in which they added 40 relations, resulting in a total of 99,280 different pairs [10]. Moreover, other researchers have made modifications to the translated version to enhance its usefulness for evaluating non-English models. These modifications have involved either removing certain relations [10] or adding more relations, as in [24].

For Arabic, Elrazzaz et al. [25] developed an Arabic Analogy Test. Their analogy contains approximately 100,000 tuples with nine relations. Another attempt was made by Abdul-Mageed et al. [26]; they developed an analogy dataset for dialectal Arabic, encompassing Algerian, Egyptian, Lebanese, Syrian, and Tunisian dialects. This dataset covers six relations, including genders, singularity, plurality, antonyms, and verb tenses (Their analogy dataset is publicly available at https://github.com/UBC-NLP/dialex (accessed on 28 September 2024)). Furthermore, Yagi et al. [10] developed an Arabic benchmark analogy comprising 45 root words covering different patterns (https://github.com/elenagara/AREEB (accessed on 28 September 2024)). This benchmark includes five derivations for nouns and variations in masculine-feminine, feminine-singular-plural, and masculine-singular-plural. Additionally, they incorporated transformations for all pronouns, synonyms, antonyms, and hyponyms of nouns, verbs, and adjectives. Finally, the benchmark encompasses capital cities and currencies.

For the sake of reliability, we decided to use a modified translated version of the Google Analogy Test, along with the benchmark provided in [10], as it comprehensively covers various aspects of Arabic. In the adaptation of the Google Analogy Test, we removed the "city-in-state" relation, as it is irrelevant to Arabic. Additionally, we excluded the "gram1-adjective-to-adverb", "gram2-opposite", "gram3-comparative", and "gram4-superlative" relations, replacing them with suitable relations from the analogy in [10]. Furthermore, we replaced the "gram8-plural" relation with the "plural" relations from [10] as well. Moreover, given that Arabic is a morphologically rich language and certain words can have multiple synonyms, we conducted the test by retrieving the top five most related words.

Furthermore, we observed that some correct answers appeared from the models with the Arabic enclitic definite article called "AL altaareef", which means "the" in English. Since the presence of this special prefix does not affect the meaning of the returned word, the answer should be considered correct with or without it.

Therefore, we used a function that returns all answers without the prefix. The main purpose of this function is to help the analogy test match the returned answers with the existing answers in the analogy precisely.

5.2. Sentiment Analysis

Sentiment analysis (SA), or opinion mining (OM), is a type of analysis that classifies the sentiment of text. SA testing, with its morphological nature, is suitable for evaluating word embedding models [27–29]. Therefore, we conducted a word-level SA.

Word-level sentiment analysis involves determining the sentiment of individual words (positive, negative, or neutral) within a text and aggregating these labels to determine the overall sentiment of the text [30]. To conduct the word-level SA, we created a sentiment lexicon named EmbeddingArabicSent that has a positive lexicon and a negative lexicon (both lexicons are available in our repository: (https://github.com/AzzahAllahim/ArabicWordEmbedding) (accessed on 28 September 2024)). We collected the words with different forms and conjugations.

Then, we conducted the test as a classification problem using an SVM classifier. The data were split, with 80% for training and 20% for testing. Finally, we computed the accuracy for each model.

5.3. Similarity Score

Since the primary goal of the embedding models is to capture the unique linguistic characteristics of Arabic, we utilize the similarity score to test if the model can determine the similarity between words. This is performed using tasks such as SemEval-2017 (semantic textual similarity task). SemEval-2017 provides pairs of Arabic sentences along with their assigned similarity scores, which are determined by human experts. In this task, the tested model is used to predict the similarity score, which is then compared to the assigned scores.

However, we found this task unsuitable for testing the efficiency of our models for three reasons. Firstly, the pairs are not well-constructed, as they include repeated terms and some pairs have sentences that cover two different areas. Secondly, the assigned similarity scores are not fully explained, as the mechanism used to calculate the score is not provided, whether it is an average score for all words or only for the important words. This absence of such calculation creates a gap in matching them with the generated scores from the models. Additionally, our main focus is to represent the words in the models; therefore, we will focus on (and test) using words instead.

To achieve this goal, we used our dataset named ArabicSynonymPairs. We constructed the dataset by compiling a list of synonyms sourced from the Arabic book [31] titled *Almojam Almofasal*, which elaborates on Arabic words and their synonyms. In addition to this book, we utilized the Almaany (https://www.almaany.com/ (accessed on 28 September 2024)) dictionary to collect the synonyms as well.

To compute the scores, we used the cosine similarity score for each pair and then calculated the average score for each model. The score ranges from 0 up to 10, where 10 represents a full match, and 0 represents no relatedness between the words.

6. Results and Discussion

In this section, we will elaborate on the results of the evaluation tests discussed in the previous section. We will review the results for each test, along with the effectiveness of each hyperparameter.

6.1. Analogy Test

We conducted individual tests for a total of 108 models using the edited version of the Google Analogy Test (The test is available in our repository: https://github.com/AzzahAllahim/ArabicWordEmbedding (accessed on 28 September 2024)). To perform the test, the most_similar() function was used for word2vec models, and the get_analogies() function was used for fastText models. Table 5 displays the overall best score for each model based on their architecture. Overall, the models trained using the Wikipedia corpus outperformed other models for both Word2vec and FastText, with best analogy scores of 80 and 90, respectively. In the following, we analyze both the Word2vec and FastText results and performances.

Table 5. Analogy best scores.

Architecture	Model	Best Score
Word2vec	Watan Corpus-CBOW	40
	Watan Corpus-Skip-gram	39
	Books Corpus-CBOW	45
	Books Corpus-Skip-gram	51
	Wiki Corpus-CBOW	84
	Wiki Corpus-Skip-gram	**86**
FastText	Watan Corpus - BOW	34
	Watan Corpus - Skip-gram	40
	Books Corpus - BOW	45
	Books Corpus - Skip-gram	49
	Wiki Corpus - BOW	80
	Wiki Corpus - Skip-gram	**90**

6.1.1. Word2vec Models

More specifically, for the Word2vec models, Figure 5 illustrates the overall results of the analogy test for the 54 Word2vec models.

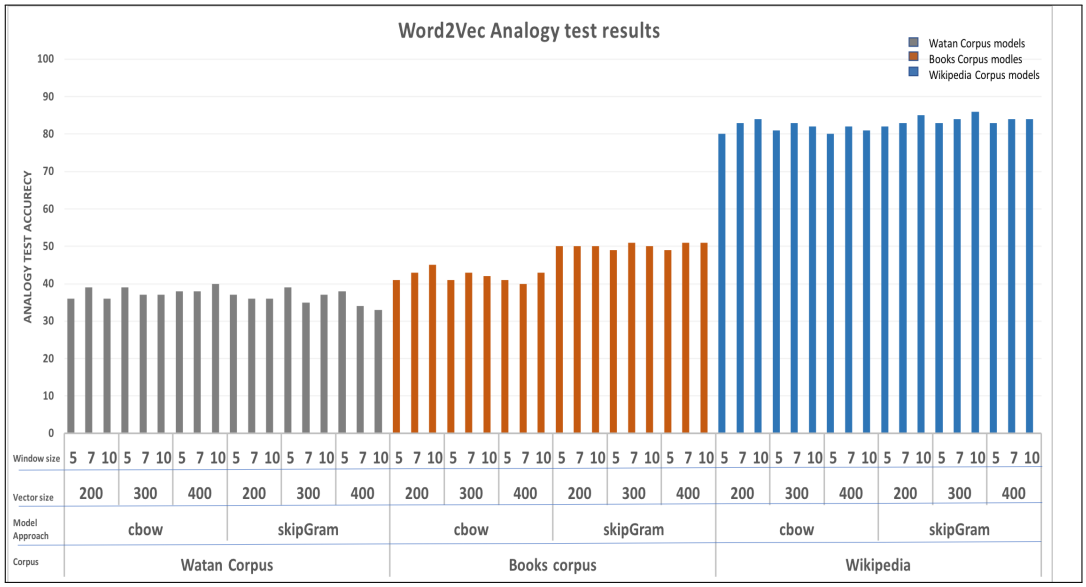


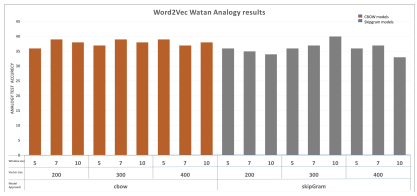
Figure 5. Word2Vec models analogy test results.

We can observe that the Wiki corpus models remarkably outperformed all the models from the other corpora. Its best analogy test score is 86, while the Watan and Book corpus scores are 40 and 51, respectively. Additionally, the Book corpus models outperformed the Watan corpus models. Below, we will demonstrate the effectiveness of each hyperparameter for each corpus. Regarding the Watan corpus, for both the CBOW and skip-gram models, the analogy test results are relatively consistent across different vector sizes, as illustrated in Figure 6a. There is no clear trend of improvement with increasing vector size. Furthermore, the choice of model (CBOW or skip-gram) also does not seem to have a significant impact on the analogy test results in this analysis.

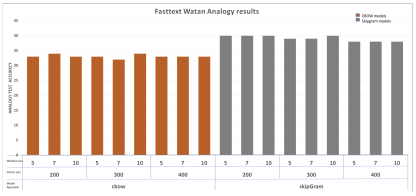
For the Books corpus, as illustrated in Figure 6c, the impact of vector size and window size on the analogy test results is relatively small. Changing the vector size from 200 to 400 and the window size from 5 to 10 does not lead to significant changes in performance in most cases. The skip-gram models consistently performed slightly better than the CBOW models for all window sizes and vector sizes in terms of the analogy test results. The difference in performance is more noticeable for larger window sizes (7 and 10) than for a smaller window size (5). The choice of vector size seems to have a more pronounced effect on the CBOW models than on the skip-gram modes. For CBOW, larger vector sizes show a slight improvement or decline in performance depending on the window size, while for skip-gram, the analogy test results remain relatively stable across vector sizes.

For the Wiki models, the results are depicted in Figure 6e, showcasing the association between the results and window size values. For vector size, increasing the vector size tends to yield improved analogy test results for both the CBOW and skip-gram models across all window sizes. This observation implies that larger vector sizes enhance the model's ability to capture semantic relationships, thereby enhancing its performance on the analogy test. On the other hand, The impact of window size on analogy test results varies across models and vector sizes. Notably, there is no discernible pattern that conclusively demonstrates the effect of window size in comparison to vector size. In most scenarios, the skip-gram models outperform the CBOW models in terms of analogy test results, particularly with larger vector sizes. This trend remains consistent across different window sizes. Indeed, the best analogy test performances for each model and window size combination were attained using specific vector sizes. For instance, the skip-gram model with a vector size of 300 achieves the highest

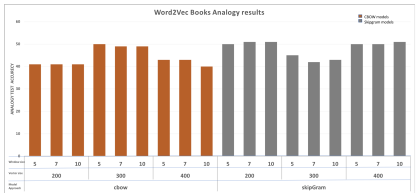
results for a window size of 10, at 86%, while the CBOW model with the same vector size achieves its peak results for window sizes 5 and 7, at 81% and 83%, respectively.



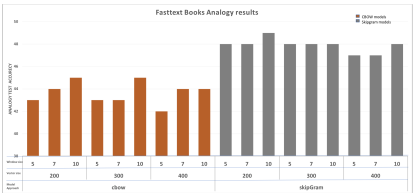
(**a**)



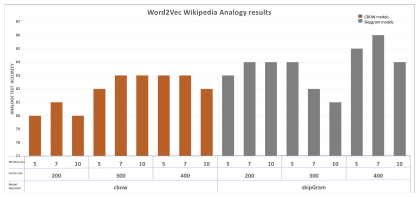
(**b**)



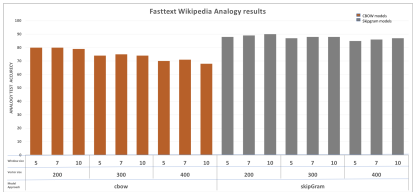
(**c**)



(**d**)



(**e**)



(**f**)

Figure 6. Model analogy test performance based on the window sizes of 5, 7, and 10 against the vector sizes 200, 300, and 400 for the CBOW and skip-gram approaches. (**a**) Word2Vec models analogy test results for the Watan corpus. (**b**) FastText models analogy test results for the Watan corpus. (**c**) Word2Vec models analogy test results for the Book corpus. (**d**) FastText models analogy test results for the Books corpus. (**e**) Word2Vec models analogy test results for the Wiki corpus. (**f**) FastText models analogy test results for the Wiki corpus.

6.1.2. FastText Models

For the FastText models, Figure 7 shows the overall results of the analogy test of 54 word2vec models. Generally, the FastText models showed more stability in performance than the Word2Vec models. Moreover, similar to the Word2vec models, the Wiki corpus models outperformed all the models of the other corpora. Its hits score had an average score of 87. Generally, the skip-gram models performed better than the CBOW models despite the used corpus. This is related to the nature of FastText, which considers the words as chunks of characters, which means that they are formed by character n-grams.

In general, similar to the Word2Vec models, the Wiki corpus models remarkably outperformed all the models of the other corpora. It scored 86 at best, while the Watan and Book corpora scored 40 and 49, respectively. The Books corpus models showed better performance than the Watan corpus models. Moreover, we analyzed the results based on the used corpus in terms of the different values of the hyperparameters.

For the Watan corpus, as illustrated in Figure 6b, both the CBOW and skip-gram models seem to show consistent performance across various vector sizes and window sizes. Consequently, the chosen approach, window size, and vector size have a limited impact on performance when using the Watan corpus.

Figure 7. FastText models analogy test results.

On the other hand, for the Books corpus, the analogy test results for both models are relatively high and consistent across different vector sizes for most window sizes. Both the CBOW and skip-gram models tend to perform well in these evaluation settings, with the analogy test results ranging from 42% to 49%. Moreover, for a window size of 7 and 10, skip-gram consistently outperforms CBOW, achieving slightly higher analogy test results, as illustrated in Figure 6d. Generally, a window size of 10 had a noticeable impact on the Book models.

For the Wiki corpus, both the CBOW and skip-gram models exhibit strong analogy test results, as illustrated in Figure 6f. Indeed, skip-gram consistently outperforms CBOW, indicating that it captures more semantic relationships and performs better on the analogy test. More specifically, both models show strong performance, with CBOW performing slightly better at a window size of 5, where it reached 79% to 80%, while the skip-gram reached 74% to 75% for different vector sizes. Moreover, skip-gram performs consistently well across vector sizes, with a notable performance improvement compared to CBOW for a window size of 7, with analogy test results ranging from 88% to 90%. In fact, it has the best analogy results across all 108 models. For a window size of 10, both the CBOW and skip-gram models' results are consistently high and range from 87% to 88% for different vector sizes. Unlike vector size, the chosen approach and window size tend to affect the results.

To summarize, for all models, training with a large vocabulary size, employing skip-gram, and utilizing a large window size can lead to good performance. Conversely, training with a small-vocabulary-size corpus can result in low analogy results and unstable performance. While Word2Vec demonstrated similar performance for CBOW and skip-gram, the FastText models achieved better results with the skip-gram approach.

Broadly speaking, there is not a widely recognized standard for evaluating the quality of analogy test outcomes. Consequently, we relied on Mikolov's research results [3] as a benchmark to gauge the outcomes of our own work.

Overall, our work achieved high scores for the Word2Vec and FastText models with 86% and 90% accuracy, respectively. This achievement outperforms the work of Chaimae et al. [9], where their best score was around 60% accuracy. Additionally, we applied the same test using the Aravec model [11], which achieved an accuracy of 79%. Table 6 presents the overall results compared with previous Arabic word embedding models. Table 7 presents the best scores of Sentiment analysis.

Table 6. Analogy test results of our work vs. previous works.

Model	Score
Azroumahli et al. [9]	60%
Aravec [11]	79%
Our model-FastText approach	**90%**
Our model-Word2Vec approach	**86%**

Table 7. Sentiment analysis best scores.

Model	Best Scores
Word2vec-Watan Corpus	91
Word2vec-Book Corpus	79
Word2vec-Wiki	85
FastText-Watan Corpus	94
FastText-Book Corpus	96
FastText-Wiki	99

6.2. Sentiment Analysis

We conducted sentiment analysis (SA) for all 108 different models, as mentioned earlier in Section 5.2. We tested the models for SA for both the Word2vec and FastText models as well.

Generally, both the Word2vec and FastText models provided good performance, with FastText significantly outperforming all Word2vec models. In particular, as illustrated in Figure 8, FastText with the CBOW architecture and Wikipedia corpus scored the best accuracy, which is 99%.

On the other hand, the Word2vec models provided unstable performance among the different models, as illustrated in Figure 9. Yet, they represented a good score with 91% accuracy with the Watan Corpus. In the following, we analyze the effect of the different hyperparameters for both the Word2vec and FastText models.

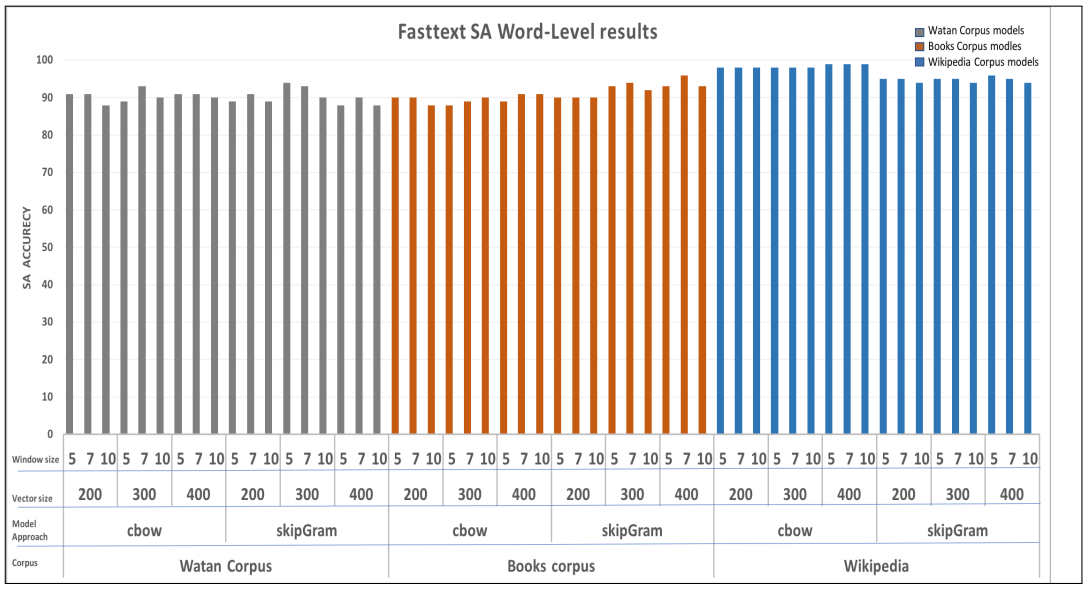


Figure 8. FastText models sentiment analysis results.

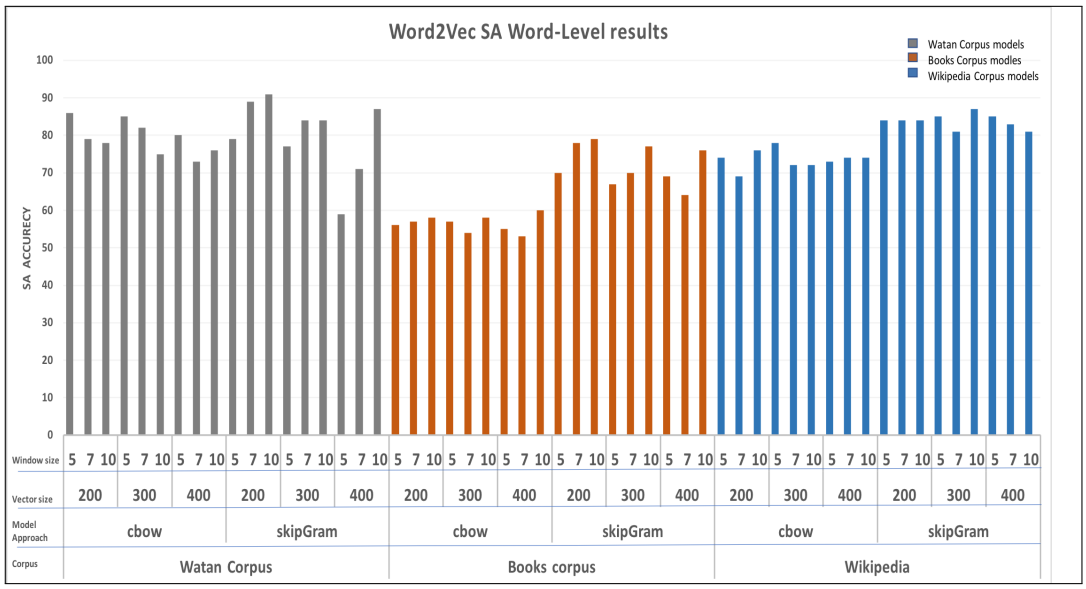
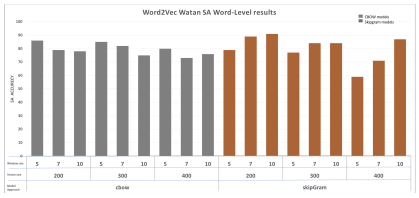


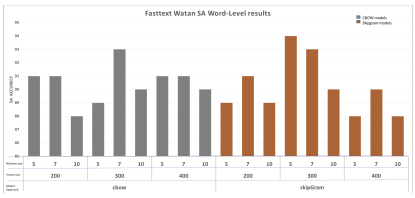
Figure 9. Word2Vec models sentiment analysis results.

6.2.1. Word2vec Models

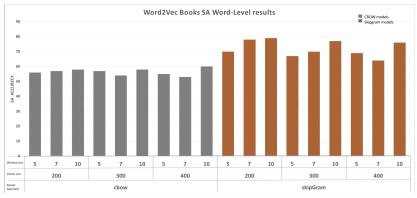
Generally, the Word2vec models that were trained using the Watan corpus presented the best performance. Moreover, for the Watan models, as shown in Figure 10a, the skip-gram models performed better than the CBOW models.



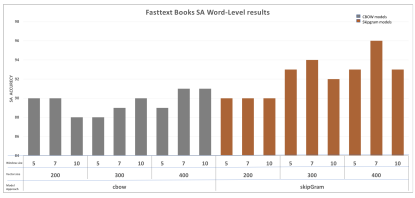
(**a**)



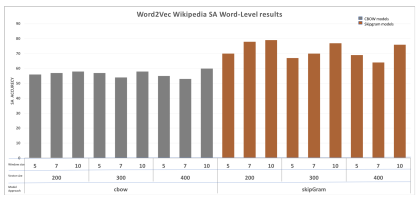
(**b**)



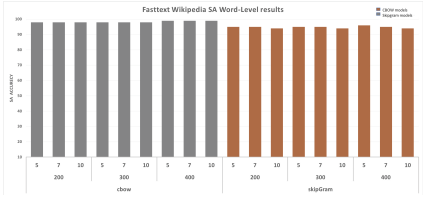
(**c**)



(**d**)



(**e**)



(**f**)

Figure 10. SA accuracy of the models based on window sizes of 5, 7, and 10 against the vector sizes 200, 300, and 400 for the CBOW and skip-gram approaches. (**a**) Word2Vec models SA results for the Watan corpus. (**b**) FastText models SA results for the Watan corpus. (**c**) Word2Vec models SA results for the Book corpus. (**d**) FastText models SA results for the Books corpus. (**e**) Word2Vec models SA results for the Wiki corpus. (**f**) FastText models SA results for the Wiki corpus.

In fact, the best results for the Watan corpus were achieved by the skip-gram model with a vector size of 200 and a window size of 10. The CBOW models seemed to perform worse with higher vector sizes. There is no clear effect of the hyperparameter values, but the only noticeable trend is that the models' accuracy showed good scores with a vector size of 200. Perhaps the reason behind this effect is that Watan has a small vocabulary size; thus, using a smaller window size can provide a more meaningful vector representation for the words that exist in the vocabulary.

For the Books models, the skip-gram models generally outperformed the CBOW models, as shown in Figure 10c. Indeed, the CBOW models' results did not show any evident effect regarding different hyperparameter values. On the other hand, the skip-gram models showed a visible pattern, with their accuracy increasing as the window size increased. The skip-gram models provided the best accuracy with a window size of 10 and a vector size of 200.

In general, the Wiki models performed well in most cases. Although the CBOW models had lower accuracy than the skip-gram models, their performance was stable among different vector sizes. The CBOW models' performance tends to improve slightly with the increase in the window size value, but it decreases as the vector size values increase. Overall, similar to the Books models, the skip-gram models have the best performance with a vector size of 200.

Overall, for the Word2vec models, it appeared that the corpus size does not have a huge effect on model performance.

6.2.2. FastText Models

For the FastText models, most of the models performed well in SA, with stable performance across different models. The Wiki models had the best accuracy scores, as illustrated in Figure 8.

Moreover, for the Watan models, as shown in Figure 10b, the CBOW models showed steady performance, where the small window sizes provided the best scores within the same vector size. However, both the CBOW and FastText models presented their best performances with a vector size of 300.

On the other hand, by increasing the corpus size, the FastText models seemed to have a more stable performance. As is shown for the Books corpus models in Figure 10d, most of the skip-gram models outperformed the CBOW models. Models with a window size of 7 for both the skip-gram and CBOW models had the best scores. Overall, the skip-gram model with a vector size of 400 presented a score of 96% accuracy.

Among all models, the FastText models with the Wiki corpus presented outstanding scores. All the models scored above 90% accuracy. The CBOW models, in particular, performed the best, with a score of 99% accuracy.

It is worth noting that for all corpora, the increase in the vector size had a negative effect on the accuracy, with the models scoring the best results at a vector size of 200. There was no clear pattern when using different window sizes. On the other hand, the CBOW models outperformed the skip-gram models, as shown in Figure 10f. The Watan models presented the worst results among all models, likely due to the fact that they were trained using a smaller vocabulary size.

Overall, the models presented good performances for word-level SA, with FastText providing the best results. Additionally, using a small vector size and a high vocabulary size corpus could lead to improved results. In general, our Word2vec models scored 91%, while the FastText models achieved 99% accuracy. As illustrated in Table 8, we also applied the test to the Aravec [11] model since it is available. Our models outperformed the Aravec [11] model, which achieved 90% accuracy. Unfortunately, the Azroumahli et al. [9] model is not available; thus, we could not perform our test on it in order to make a fair comparison.

Table 8. Sentiment analysis results of our work vs. previous works.

Model	Score
Aravec [11]	90%
Our model-FastText approach	**99%**
Our model-Word2Vec approach	**91%**

6.3. Similarity Score

In this study, the similarity test was applied to the Word2Vec and FastText models.

As shown in Figure 11, the FastText models achieved an optimal similarity score of approximately 8 out of 10. The models displayed a clear performance trend, with scores consistently improving as the vocabulary size decreased. Notably, the Wiki skip-gram models outperformed the CBOW models in the majority of cases. Additionally, the models consistently yielded their best results when employing a vector size of 200 with a higher window size. This indicates that a smaller vector size can have a positive impact on the models' performance.

Conversely, the Word2Vec models displayed distinct performance characteristics, as illustrated in Figure 12. They achieved an optimal score of approximately 7.3. Notably, models trained on the Wiki corpus consistently surpassed all others in performance. Unlike FastText, the Word2Vec models delivered their best results when utilizing the CBOW architecture in most cases. Additionally, employing a smaller window size also had a positive impact on their performance.

The strong performance observed in the Wiki corpus models can be attributed to the fact that Wiki models have a larger vocabulary size. This particular factor seems to have had a notable impact on the FastText models trained on the Books corpus. Hence, reducing the window size and vector size had an impact on the outcomes for both the Word2Vec and FastText models. The size of the vocabulary affected the FastText models, whereas the quality of the vocabulary affected the FastText models.

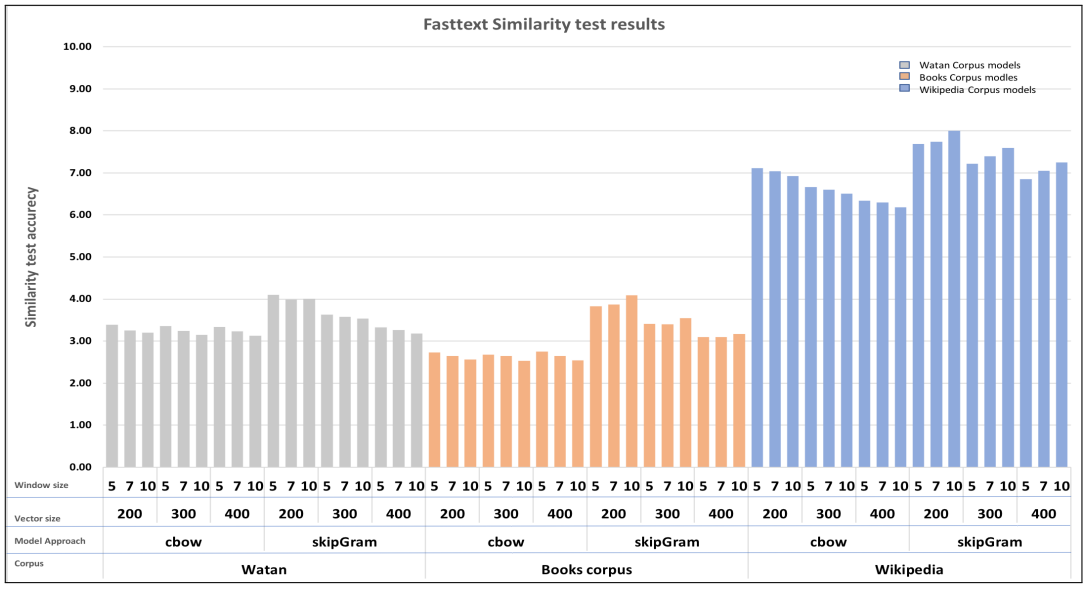


Figure 11. FastText models similarity test results.

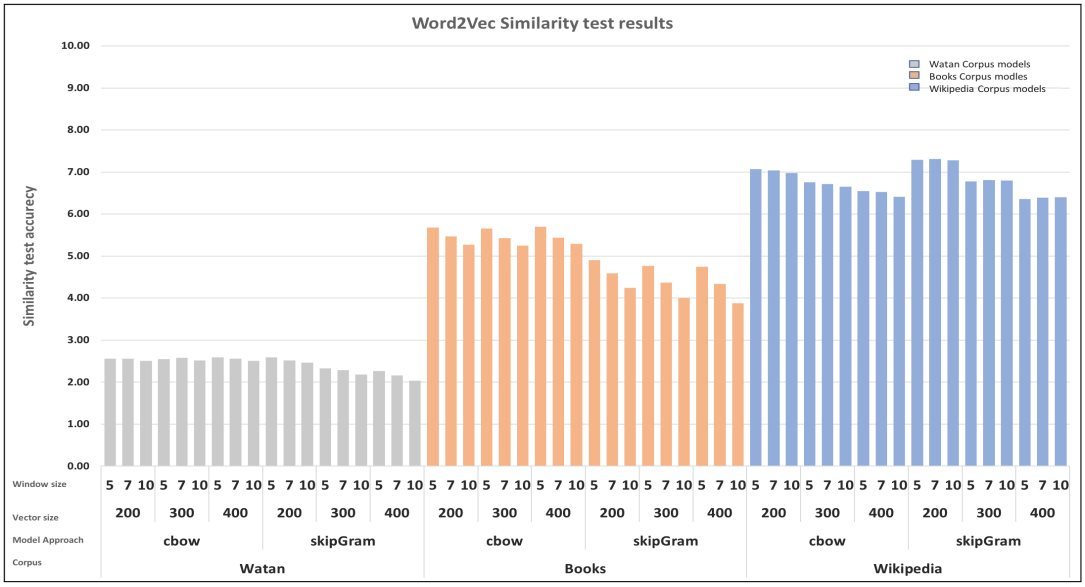


Figure 12. Word2Vec models similarity test results.

To conclude, our observations revealed that the size of the corpus and the values of the hyperparameters had varying impacts on each test. In the analogy test, we found that employing a large vocabulary size had a positive influence on the results. Additionally, the FastText models with skip-gram approaches proved effective in solving analogy test questions. For sentiment analysis, we discovered that vocabulary size played a crucial role. Furthermore, for the FastText models, applying CBOW architecture led to improved accuracy, whereas for Word2Vec, applying skip-gram architecture enhanced accuracy. Additionally, smaller window size had a positive effect on accuracy. Finally, in the similarity score test, the FastText models delivered the highest scores with the Wiki corpus and skip-gram architecture. Smaller window sizes and smaller vector sizes positively impacted the results. Hence, since the impact of the hyperparameters varies for each test, we can infer that there is no universally superior hyperparameter setting that guarantees high accuracy across all NLP tasks. This implies that each test has its own specific criteria that need to be met [1].

7. Conclusions

This research focused on developing Arabic word embeddings. Our approach involved creating 108 different models, each with unique combinations of hyperparameters and corpora. To build our corpora, we leveraged available online resources, as well as Arabic Books. We utilized the Word2Vec and FastText libraries in our modeling process.

A series of evaluations were conducted to assess the performance of these models. Notably, we achieved a 90% accuracy rate in the analogy test. In the sentiment analysis tests, our models achieved approximately 99% accuracy. Furthermore, in the similarity score test, some of our models attained an average similarity score of 8 for all pairs in the dataset.

Our observations revealed that the size of the corpus and the values of the hyperparameters had varying impacts on each test:

- In the analogy test, it has been shown that employing a large vocabulary size had a positive influence on the results. Additionally, the FastText models with skip-gram approaches proved to be effective in solving analogy test questions.
- For sentiment analysis, we discovered that vocabulary size played a crucial role. Furthermore, for the FastText models, applying CBOW architecture led to improved accuracy, whereas for Word2Vec, applying skip-gram architecture enhanced accuracy. Moreover, smaller window size has a positive effect on accuracy.
- Finally, in the similarity score test, the FastText models delivered the highest scores. Smaller window sizes and smaller vector sizes positively impacted the results.

For future work, we will consider building models specifically for different Arabic dialects.

Author Contributions: Conceptualization, A.A. and A.C.; methodology, A.A.; software, A.A.; validation, A.C.; formal analysis, A.A. and A.C.; writing—original draft preparation, A.A.; writing—review and editing, A.C.; visualization, A.A.; supervision, A.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Lai, S.; Liu, K.; He, S.; Zhao, J. How to Generate a Good Word Embedding? *IEEE Intell. Syst.* **2017**, *31*, 5–14. [CrossRef]
2. Maas, A.L.; Daly, R.E.; Pham, P.T.; Huang, D.; Ng, A.Y.; Potts, C. Learning Word Vectors for Sentiment Analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 19–24 June 2011; pp. 142–150.
3. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *Proc. Workshop ICLR* **2013**, *2013*.
4. Salama, R.A.; Youssef, A.; Fahmy, A. Morphological Word Embedding for Arabic. *Procedia Comput. Sci.* **2018**, *142*, 83–93. [CrossRef]
5. Joulin, A.; Grave, E.; Bojanowski, P.; Douze, M.; Jégou, H.; Mikolov, T. FastText.zip: Compressing text classification models. *arXiv* **2016**, arXiv:1612.03651.
6. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching Word Vectors with Subword Information. *Trans. Assoc. Comput. Linguist.* **2017**, *5*, 135–146. [CrossRef]
7. Grave, E.; Bojanowski, P.; Gupta, P.; Joulin, A.; Mikolov, T. Learning Word Vectors for 157 Languages. *arXiv* **2018**, arXiv:1802.06893.
8. Azroumahli, C.; El Younoussi, Y.; Achbal, F. An Overview of a Distributional Word Representation for an Arabic Named Entity Recognition System. In Proceedings of the Ninth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2017), Marrakech, Morocco, 11–13 December 2017; pp. 130–140. [CrossRef]
9. Azroumahli, C.; Rybinski, M.; Younoussi, Y.; Montes, J. Comparative study of Arabic Word Embeddings: Evaluation and Application. *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.* **2020**, *12*, 349–362.
10. Yagi, S.; Elnagar, A.; Fareh, S. A benchmark for evaluating Arabic word embedding models. *Nat. Lang. Eng.* **2022**, *9*, 978–1003.
11. Soliman, A.B.; Eissa, K.; El-Beltagy, S.R. AraVec: A set of Arabic Word Embedding Models for use in Arabic NLP. *Procedia Comput. Sci.* **2017**, *117*, 256–265. [CrossRef]
12. Mikolov, T.; Grave, E.; Bojanowski, P.; Puhrsch, C.; Joulin, A. Advances in Pre-Training Distributed Word Representations. *arXiv* **2017**, arXiv:1712.09405.
13. Naili, M.; Chaibi, A.H.; Ben Ghezala, H.H. Comparative study of word embedding methods in topic segmentation. *Procedia Comput. Sci.* **2017**, *112*, 340–349. [CrossRef]
14. Pennington, J.; Socher, R.; Manning, C. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543. [CrossRef]
15. Athiwaratkun, B.; Wilson, A.G.; Anandkumar, A. Probabilistic FastText for Multi-Sense Word Embeddings. *arXiv* **2018**, arXiv:1806.02901.
16. Oueslati, O.; Cambria, E.; HajHmida, M.B.; Ounelli, H. A review of sentiment analysis research in Arabic language. *Future Gener. Comput. Syst.* **2020**, *112*, 408–430. [CrossRef]
17. Kayraldeen, A. Ketab Alaalam. 1926. Available online: https://www.kutubltd.com/book_year/1926/ (accessed on 28 September 2024).
18. Abbas, M. Watan 2004 Corpus. 2004. Available online: https://metatext.io/datasets/watan-2004-corpus (accessed on 28 September 2024).
19. Attardi, G. WikiExtractor. 2015. Available online: https://github.com/attardi/wikiextractor (accessed on 28 September 2024).
20. Chen, Z.; He, Z.; Liu, X.; Bian, J. Evaluating semantic relations in neural word embeddings with biomedical and general domain knowledge bases. *BMC Med. Inform. Decis. Mak.* **2018**, *18*, 65. [CrossRef]
21. Zahran, M.A.; Magooda, A.; Mahgoub, A.Y.; Raafat, H.; Rashwan, M.; Atyia, A. Word Representations in Vector Space and their Applications for Arabic. In *Proceedings of the Computational Linguistics and Intelligent Text Processing*; Gelbukh, A., Ed.; Springer: New York, NY, USA, 2015; pp. 430–443.
22. Dahou, A.; Xiong, S.; Zhou, J.; Haddoud, M.H.; Duan, P. Word Embeddings and Convolutional Neural Network for Arabic Sentiment Classification. In Proceedings of the COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, 11–16 December 2016; pp. 2418–2427.

23. Gladkova, A.; Drozd, A.; Matsuoka, S. Analogy-based detection of morphological and semantic relations with word embeddings: What works and what doesn't. In Proceedings of the NAACL Student Research Workshop, San Diego, CA, USA, 13–15 June 2016; pp. 8–15. [CrossRef]
24. Khusainova, A.; Khan, A.; Rivera, A.R. SART - Similarity, Analogies, and Relatedness for Tatar Language: New Benchmark Datasets for Word Embeddings Evaluation. In *Proceedings of the Computational Linguistics and Intelligent Text Processing*; Gelbukh, A., Ed.; Springer: New York, NY, USA, 2023; pp. 380–390.
25. Elrazzaz, M.; Elbassuoni, S.; Shaban, K.; Helwe, C. Methodical Evaluation of Arabic Word Embeddings. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Vancouver, BC, Canada, 30 July–4 August 2017; pp. 454–458. [CrossRef]
26. Abdul-Mageed, M.; Elbassuoni, S.; Doughman, J.; Elmadany, A.; Nagoudi, E.M.B.; Zoughby, Y.; Shaher, A.; Gaba, I.; Helal, A.; El-Razzaz, M. DiaLex: A Benchmark for Evaluating Multidialectal Arabic Word Embeddings. In Proceedings of the Sixth Arabic Natural Language Processing Workshop, Kyiv, Ukraine (Virtual), 19 April 2021; pp. 11–20.
27. Brahimi, B.; Touahria, M.; Tari, A. Improving sentiment analysis in Arabic: A combined approach. *J. King Saud Univ.-Comput. Inf. Sci.* **2019**, *33*, 1242–1250. [CrossRef]
28. Ibrahim, H.S.; Abdou, S.M.; Gheith, M. Sentiment Analysis For Modern Standard Arabic And Colloquial. *arXiv* **2015**, arXiv:1505.03105. [CrossRef]
29. Iqbal, F.; Hashmi, J.M.; Fung, B.C.M.; Batool, R.; Khattak, A.M.; Aleem, S.; Hung, P.C.K. A Hybrid Framework for Sentiment Analysis Using Genetic Algorithm Based Feature Reduction. *IEEE Access* **2019**, *7*, 14637–14652. [CrossRef]
30. Nassif, A.B.; Elnagar, A.; Shahin, I.; Henno, S. Deep learning for Arabic subjective sentiment analysis: Challenges and research opportunities. *Appl. Soft Comput.* **2021**, *98*, 106836. [CrossRef]
31. Terad, M. Almojam Almofasal; Dar Alkotob Alarabiah. 2011. Available online: https://books.google.co.kr/books/about/%D8%A7%D9%84%D9%85%D8%B9%D8%AC%D9%85_%D8%A7%D9%84%D9%85%D9%81%D8%B5%D9%84_%D9%81%D9%8A_%D8%A7%D9%84%D9%85%D8%AA.html?id=6eVhDwAAQBAJ&redir_esc=y (accessed on 28 September 2024).