International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 12 (2020) pp. 349-362

Received: 7 May, 2020; Accepted: 19 Oct., 2020; Published: 31 Oct., 2020

Comparative study of Arabic Word Embeddings: Evaluation and Application

AZROUMAHLI Chaimae[1], Maciej Rybinski[2], El Younoussi Yacine[3], Jos éF. Aldana Montes[4]

[1] ENSA Tetuan, Abdel Malek Essa âdi University, Morocco
c.azroumahli@uae.ac.ma

[2] Khaos Research, Universidad de Malaga, Spain
maciek.rybinski@lcc.uma.es

[3] ENSA Tetuan, Abdel Malek Essa âdi University, Morocco
yacine.elyounoussi@uae.ac.ma

[4] Khaos Research, Universidad de Malaga, Spain
jfam@lcc.uma.es

***Abstract*: Word Embeddings models have achieved impressive results for a variety of NLP tasks in the last few years which, consequently, led to the increasing interest in creating more adequate word representations to capture semantic and syntactic features for certain languages. This study aims to train different Arabic Word Embeddings models in a supervised framework and to investigate the impact that the models' hyper-parameters have on downstream Arabic NLP tasks and applications. In this paper, we present the cleaning and the pre-processing steps followed to create three different training datasets. We provide a detailed description of the steps we followed for creating 180 different Word Embeddings models using Word2Vec and CBOW. To evaluate the quality of the Word Embeddings, we perform several extrinsic and intrinsic evaluation methods. The preliminary results show that these models can create meaningful Word Embeddings despite the higher-logical complexity of the Arabic language. We have concluded that the source of the training dataset is significant to the type of information captured by the model. Moreover, the hyper-parameters of training architecture and the nature of an NLP task is significant to its accuracy.**

***Keywords*:** Word Embeddings, Evaluation methods, NER, Document Classification, POS-tagging, Sentiment Analysis.

I. Introduction

Word representations started as a mapping technique for high-dimensional one-hot vectors that encode words to lower-dimensional dense vectors, without any notion of similarity between words. This technique made it easier for NLP researchers to process natural languages, as the majority of classification and deep learning algorithms don't take raw texts as inputs. The introduction of the distributional hypothesis to these word representations [1], [2], made them capable of achieving impressive results for certain natural languages understanding tasks. Nowadays, Word Embeddings became an efficient method for learning meaningful dense vector representations of words. They proved to be very useful for various NLP applications. The vector representation addresses the problem of dimensionality and improves generalization by mapping words with related semantic and syntactic meanings near each other in a continuous space.

The benefits of using these Word Embeddings models have been highlighted in many natural languages other than English, like Portuguese [3], Italian [4], and even for languages known by their rich morphology and a complex syntax like Persian [5] and Arabic [6]. The objective of this paper is to provide helpful resources for Arabic NLP applications while investigating the effect of two hyper-parameters on Word Embeddings' intrinsic evaluation tests and NLP applications. Specifically, we describe the steps followed for creating 180 different models using different combinations of the contextual window, the vectors dimension, and datasets. We investigate the impact of each algorithm's hyper-parameters by tuning the vector's dimension and the contextual window. We compare the result models using several extrinsic and intrinsic evaluations.

The rest of this paper is structured as follow, we start by highlighting the data pre-processing steps, and give a brief description of CBOW, Skip-Gram, Glove and Fast-text, the models used for this study. We proceed by illustrating the experiment setups we followed to create the different models. Then, we describe how we assess their quality using different intrinsic and extrinsic evaluation methods. We present and discuss the results on various evaluation test, and finally, we conclude by introducing our perspective for future works.

II. Building Arabic Word Embeddings

A. Background: Word Embeddings models

Word Embeddings are words' representation in n-dimensional space with vectors of real-valued numbers. These numbers are learned using unsupervised techniques considering as inputs a word and its context in a large non-annotated corpus. The resulted vector representations can carry the semantic and the syntactic properties of words. There is a wide range of state-of-the-art approaches that have been created to exploit the idea

of the context of words. In this sub-section, we describe the four approaches we adapted for this study to build 180 different Word Embeddings models: Word2vec's Skip-Gram and CBOW, Glove, and FastText.

1) Word2vec: Skip-Gram and CBOW

Word2vec is a family of algorithms known by the fast log-linear training that captures semantic information; these algorithms pre-train a single projection matrix $W \in \Re^{d \times |V|}$ where d is the embedding dimension and V is the vocabulary. They build embeddings by maximizing the likelihood of word prediction of their context and vice versa [2].

Two different training methods were defined by Word2vec: 1) *Skip-Gram* that uses a word as input to predict the surrounding ones in a fixed window; 2) *Continuous bag-of-words* (CBOW) where a target word's context in a surrounding fixed window is a sequence of inputs, and the target word is the output. In the original Skip-Gram and CBOW models introduced by Mikolov and his colleagues, they used a computationally efficient approximation of the full Softmax to evaluate the gradient i.e. *the hierarchical Softmax (HS)*, where the vocabulary is represented as a Huffman binary tree that results in evaluating only $log_2(W)$ of the output nodes in the neural network instead of evaluating all of the nodes W. Later on, they represented another extension to their models to improve the quality of the frequent words' vectors and speed up the training process. In this extension, they alternated the *hierarchical Softmax* with *negative sampling* (NS) which is build based on the idea of the noise contrastive estimation, and that a good model should differentiate fake signal from the real ones by the means of logistic regression [7].

In this work we experiment with both of these approximation algorithms, to investigate the claim of HS being better with infrequent words and NS with frequent words and low dimensional vectors.

2) Glove: Global vectors

Glove is a global log-bilinear regression model with a weighted least-squares objective, trained on the global corpus statistics of word occurrences to produce linear directions of meanings [8]. This model is based on the hypothesis that suggests that the ratio of word-word co-occurrence probabilities encode the meaning of the words. This model learns the word representations starting by studying the ratio in (1) of the co-occurrence probabilities of two words, i and j with various probe words k (i.e. Context words). This ratio is large for context words related to i and small for context words related to j, while it is close to 1 when the context word is related to both words. (1) encodes the relationship between three words while w_i, w_j are the vector representation for the words i *and* j. The model is trained using a gradient descent algorithm.

$$\frac{P_{ik}}{P_{jk}} = F(w_i, w_j, \widetilde{w}_k) \quad (1)$$

3) FastText: CBOW and Skip-Gram

FastText is a library proposed by the Facebook research team for both learning word representation and sentence classification [9]. On the contrary to other representation (i.e. Word2vec, CBOW) where Embeddings are associated to every single word as a unit, FastText assumes that words are formed by character n-grams, and words are a summarization of these representations. This difference leads FastText models to generate better word representation for rare words since their character n-grams are shared with other non-rare words. Further, it can generate vectors for words that don't appear in the training corpus from its character n-grams.

B. Data collection

The Arabic speech community exhibits a phenomenon known as the DIGLOSSIA situation, where different varieties of the same language are used by the same community, and each one of these varieties is used for a specific purpose [10]. There are three main Arabic varieties: The Classical Arabic (CA) or the language usually used in religious and literature contexts, it is fully structured and vowelized [11]. The Modern Standard Arabic (MSA), which is the official language used in education, media and formal communications across the different Arabic speaking countries, it is based on the CA's syntax and morphology, but it tends to have a more modern vocabulary and "loanwords" [12]. And finally, the Arabic colloquial dialects (AD), or the language used in daily informal conversations, it has no orthographic standards so one word can be written in different forms [13]. Additionally, the AD variety can vary from a region to another across the Arabic countries.

To provide the most efficient study of Arabic Embeddings models', we needed to collect data from different text domains to get different Arabic varieties. We have focused on the MSA and the AD varieties since CA is mostly used in religious scriptures.

We collected a large corpus from two main web resources to obtain a multi-genre corpus representative of Arabic. The first source being the online encyclopaedia; Wikipedia that provides more than 810k Arabic articles. We used the function-based harvesting algorithm provided by Wikipedia library[1] to create our Wikipedia corpus.

Our second corpus contains a variety of Arabic dialects, such as Egyptian, Gulf, Moroccan, Tunisian, Algerian, Levantine... To create such a corpus, and since, as [14] explained, the language used in social media is known to be highly dialectal, we went with two of the most famous social media sites as our second source. 1) Twitter, which is the most targeted social media platform for Arabic Dialects analysis due to its easiness in gaining access to a wider range of Arabic tweets using twitter API [15]. 2) Facebook, because, as the study conducted in [16] shows, the middle eastern use both these social media platforms, but the North African tend to utilize Facebook more than Twitter [16], and the goal was to analyse Word Embeddings for all the varieties of Arabic Dialects.

The process of creating a corpus using Twitter API involves three main steps as Figure 1 demonstrates; Preparing the python authentication using Tweepy[2], streaming and filtering the Arabic tweets from Arabic users where only the live tweets that contain Arabic script were extracted. The same methodology was followed to collect Facebook comments using Facebook API and an Excel Add-In[3]; once the Data connection was configured, we specified the unbiased

[1] https://pypi.org/project/wikipedia/

[2] An open source python Library used mainly to access the Twitter API.

[3] A tool that sets a connection with live Facebook data with Microsoft Excel, https://download.cnet.com/Excel-Add-In-for-Facebook/3000-2065_4-76476610.html

Facebook pages and groups that will be populated with live Facebook data. Table 1 shows the statistics of the collected data.

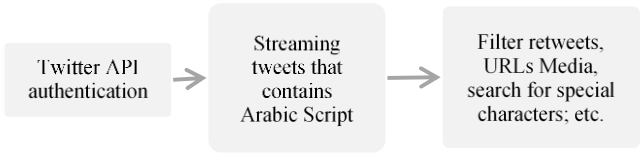


Figure 1. The steps of creating a corpus using Twitter API

Source	Wikipedia	Facebook	Twitter	Social media
Targeted Arabic variety	MSA	MSA & North African dialects	MSA & Middle Eastern dialects	MSA & Arabic Dialects
Types' count	154 811	92 295	74 739	137 756

Table 1. The statistics of the collected data

C. Pre-processing of the training corpus

Arabic is classified as a highly inflected language; it is known to have a rich morphology and a complex syntax [17]. To create our Word Embeddings for Arabic, we applied several pre-processing steps to the obtained corpora respecting Arabic characteristics.

Numbers, URLs, and social media users' mentions (@user) for Facebook and Twitter were excluded by removing all the Non-Arabic letters. This step included the removal of diacritical marks as well. These diacritics were removed because they are used optionally, so, their existence in a corpus will be inconsistent.

The characters were normalized by unifying the shape of some Arabic letters to make the text in a consistent form as suggested by [18]. Table 2 shows the normalization cases we followed. The text was also tokenized into different words and sentences using the NLTK library [19]. We further pre-processed the data by disregarding all the Arabic stop words based on the list provided in [20].

Letters	Normalization
{أ،آ،ٱ،إ،ا}	{ا}
{ة، ه}	{ه}
{ي،ى}	{ى}

Table 2. Normalization cases

D. Experiments setup

In this work, we investigate the relationships of two hyper-parameters; the vector dimension and the contextual window. We explore their impact by training Word Embeddings for 24 different parameterizations using various state of the art approaches; CBOW, Skip-Gram and Glove architectures, on 4 training data sets. Then investigate their performances on different downstream Arabic NLP applications.

To create the Word Embeddings models, we used the Genism implementation of word2vec to train the CBOW and Skip-Gram models, the Glove-Python implementation to train the Glove models, and two of the available FastText models trained using CBOW and Skip-Gram architectures on Arabic crawled Wikipedia articles. We set Word2vec and Glove models on various dimensions (3,5,7,9), and various contextual window lengths (200,300,400) to investigate the relationships of these two hyper-parameters. We choose 10 as a minimum count of words to avoid the increasing number of misspelt words that exist in social media content. As for the rest of the hyper-parameters, we used the default ones included in the packages. Table 3 illustrates the Data-set's sources and the hyper-parameters values used to train the models.

	CBOW (HS)	CBOW (NS)	Skip-Gram (HS)	Skip-Gram (NS)	Glove	FastText
Data-set's source	Wikipedia, Facebook, Twitter, Social media	Wikipedia, Facebook, Twitter, Social media	Wikipedia, Facebook, Twitter, Social media	Wikipedia, Facebook, Twitter, Social media	Wikipedia, Facebook, Twitter, Social media	Wikipedia
Contextual window	3,5,7,9	3,5,7,9	3,5,7,9	3,5,7,9	3,5,7,9	10
Vectors' dimension	200,300,400	200,300,400	200,300,400	200,300,400	200,300,400	300

Table 3. Models' hyper-parameter configurations

III. Evaluating Arabic Word Embeddings

To evaluate the trained Word Embeddings models, we used intrinsic and extrinsic evaluations. For the intrinsic evaluation, we performed the word Analogy test and a concept categorization test. As for the extrinsic evaluation, we used different NLP tasks, namely Named Entity Recognition (NER), POS tagging, Document Categorization (DC) and Sentiment Analysis (SA). The following sub-sections present a brief description of every evaluation method we used.

A. Intrinsic Evaluation

1) Word Analogy task

Word Analogy test was first introduced by the authors of [2], where the goal is to solve analogy questions by finding a term x for a given term y, so that $x : y$ is the best resemblance to a sample relationship $a : b$. This test was conducted using an enhanced and translated version of Google's word analogy test benchmarks, aided by the available benchmarks published by the authors of [21]. An example of a typical analogy relation from our benchmarks test set contains two pairs of words as is illustrated in Table 4. The benchmarks contain 71k relations and cover 11 relation types, 4 of them are semantic and 7 are syntactic (see Table 5). To compute the vector's analogies, we needed to recover the relations between the word vectors. Algorithm 1 presents the algorithm used to compute these relations and report the Word analogy accuracy.

Relation type	Word pair 1		Word pair 2	
Capitals of the world	لشبونة Lisbon	البرتغال Portugal	مدريد Madrid	اسبانيا Spain
Opposite relations	نام Slept	استيقظ Woke up	حزين Sad	سعيد Happy

Table 4. Word Analogy test-set example

Syntactic	Semantic
Comparative Plurals Pairs Vern to noun	Capital cities Common capital cities (for the Arab world) Currency Family Man Woman Nationality adjectives Opposite
#35978	#35996

Table 5. Word analogy relation types

Algorithm 1: Word Analogy computing

Input: The model (vector representations of a vocabulary), Word analogy benchmarks
Output: Word analogy accuracy
1: Function: Word analogy accuracy
2: Begin
3: accuracy=0
4: For each word analogy relation type:
5: For each two different word pairs $(\boldsymbol{a},\boldsymbol{b})\&(\boldsymbol{c},\boldsymbol{d})$ while $\boldsymbol{d}$ is the hidden word and $\boldsymbol{a}$ to $\boldsymbol{b}$ has the same relation as $\boldsymbol{c}$ has to $\boldsymbol{d}$:
6: Calculate the vector representation of the target word using the vector representations of $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ from the model: $\boldsymbol{t}[\ ] = \boldsymbol{b}[\ ] - \boldsymbol{a}[\ ] + \boldsymbol{c}[\ ]$.
7: For each word w_i existing in the vocabulary $\boldsymbol{V}$:
8: Compute the similarity between $\boldsymbol{t}[\ ]$ and the vector representation of w_i using the equation: $\boldsymbol{argmax}_{w_i \in V}\ \mathbf{cos}(\boldsymbol{w_i}[\ ], \boldsymbol{t}[\ ])$.
9: Retrieve the most similar word to the target word $\boldsymbol{t}$: $\boldsymbol{w_t} = \mathbf{min}(\boldsymbol{argmax}_{w_i \in V}\ \mathbf{cos}(\boldsymbol{w_i}[\ ], \boldsymbol{t}[\ ]))$.
10: Update the accuracy: if the word $\boldsymbol{w_t}$ is the hidden word $\boldsymbol{d}$.
11: Return Accuracy/number of relations
12: End

Algorithm 1. Word Analogy computing algorithm

2) Concept categorization

The Concept Categorization or word clustering is the task of grouping a set of nominal concepts into natural categories. We used this task as an evaluation method for the trained Word Embeddings to examine the word representations' similarities and the ability of Word Embeddings space to be clustered. With this evaluation, we test the fact that the main goal of Word Embeddings is words with similar meaning have closer vector representations, thus, tend to be closer in the continuous space. To the best of our knowledge, there is no standard categorization benchmark for Arabic. We provide a categorization data set that contains 2148 concepts clustered into 22 categories[4]. Table 6 illustrates an example of this set of concepts, and Table 7 gives the categories and the statistics of each category. To compute the categorization accuracy, we used the K-means algorithm on the obtained word representations and examined the cluster quality by calculating the purity measure. Algorithm 2 presents the algorithm used to compute the clusters, and report the purity measure as a categorization accuracy.

Categories	Concepts
/Professions / حرف	/lawyer/، محامي /lieutenant/، ملازم /Announcer/، معلن …
/Family / عائلة	/Wife/، زوجة /Aunt/، خالتي /Grandmother/، جدتي /Brother/، أخ …
/Sport/ رياضة	/Swimming/، السباحة /Shooting/، الرماية /Skiing/، التزلج …

Table 6. Concept categorization test-set examples

Category name	Count	Category name	Count
/ Musical instrument/آلة موسيقية	35	/Religious services/خدمات الدينية	62
/Furniture or part of a building/اثاث او جزء من مبنى	206	/ Sports/رياضة	44
/Precious stones/الأحجار الكريمة	28	/Family/عائلة	53
/Trees and Flowers/الأشجار والزهور	71	/Sciences/علوم	41
/Colors /الالوان	39	/ Cities & countries/مدن ودول	155
/ Crimes and weapons/الجرائم والأسلحة	169	/ Vehicles/مركبات	54
/ Human body & diseases/الجسم البشري والامراض	138	/ Dwelling, office/مسكن او مكتب	71
/Vegetables, fruits & food/الخضروات الفواكه ونكه الطعام	148	/ Profession, Craft/مهنة او حرفة	202
/ Earth & weather/الكرة الارضية والطقس	143	/ Type of fuel/نوع من الوقود	62
/ Clothing and types of fabrics/الملابس وانواع الاقمشة	112	/ Reading material/نوع من مواد القراءة	78
/Animals & reptiles/حيوانات حشرات زواحف	184	/ Distance, time/المسافة او الوقت	53

Table 7. Concept categorization test-set statistics

[4] https://github.com/AzChaimae/Categorization-dataset-for-Arabic

Algorithm 2: Concept categorization computing

Input: The model, Categorization benchmarks (Concepts and their categories)
Output: Concept categorization accuracy
1: Function: Concept categorization accuracy
2: Begin
3: set **k** the number of clusters (or categories).
4: set **N** the number of objects (Concepts)
5: get $\mathbf{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_N\}$ the vectorized representations of the concepts.
6: get **C** the predicted clusters using the K-Means algorithm from Scikit-learn library, while using the Euclidian distance as a metric.
7: get **T** the true clusters from the benchmarks
8: transform each $\boldsymbol{t}_j$ from T to its vectorized representation $\boldsymbol{v}_j$ from V
9: for each $\boldsymbol{c}_i$ from the clusters $\boldsymbol{C}$:
get $\boldsymbol{t}_j$ from $\boldsymbol{T}$: classification with the max count for $\boldsymbol{c}_i$
10: Count $\boldsymbol{M} = \sum_{i=1}^{k} \mathbf{max}|\boldsymbol{c}_i \cap \boldsymbol{t}_j|$ the number of correctly assigned vectors
11: Return Purity measure
12: End

Algorithm 2. Concept categorization computing algorithms

B. Extrinsic Evaluation: NLP tasks

The Intrinsic evaluation methods are known for being computationally cheap, but they are not adequate to study the Word Embeddings performance on downstream NLP tasks. Extrinsic evaluations measure the contribution of a word Embedding model to a specific NLP task. In this sub-section, we describe the evaluations performed on 4 NLP applications namely: Named Entity Recognition, Part-of-Speech tagging, Document Categorization and Sentiment Analysis.

1) Named Entity Recognition (NER)

NER is the problem of locating important nouns (words or phrases) in a text and classifying them into predefined semantic classes. These nouns usually hold key information in a sentence which serves as an important target for most NLP systems [22]. Our application is performed on an annotated corpus provided by AQMAR project [23], after preprocessing it following the steps described in Section II-C. This dataset contains 28 articles hand-annotated to 9 named entities, using the BIO system tags (e.g. O, B-PER, B-MIS, B-ORG, B-LOC, I-PER, I-MIS, I-ORG, I-LOC). Table 8 illustrates the statistics of the NER annotated dataset and an example of the annotated tokens. In our experiment, we used an implementation of LSTM networks from Keras library as a classification algorithm [24][25]. This algorithm computes for each word a score of the considered classes using the BIO system adopted in the annotated corpus. Furthermore, the data was split into training, validation and testing sets. The word representations of the training and validation sets were fed to an LSTM layer, followed by a sigmoid activation layer, and complied by Adam, a stochastic Optimization layer. We did not focus on optimizing the classification hyper-parameters; instead, we set a single configuration to compare between the Embedding's models.

Number of tokens	Examples the annotated tokens
57858	… ، و انتهاء بنظرية الأوتار الفائقة . … … O O O B-MIS I-MIS I-MIS O …

Table 8. Statistics of the NER annotated corpus

2) Part-of-Speech tagging (POS)

POS tagging task also called word-category disambiguation is a fundamental task in NLP applications. It is a very suitable task to evaluate the syntactic quality of Word Embeddings. We used Kalimat annotated corpus [26], that was created using the Stanford POS-Tagger to annotate 20291 articles with an average of 2500 token per article. The annotation identifies 33 Part of speeches such as Noun (NN), Plural Noun (NNS), Proper Noun (NNP), Verb (VB), Adjective (JJ), ... The input of the NER architecture is Word Embeddings where one word has one vector representation that doesn't change with its context. If we used the same architecture used for NER, the POS application will give erroneous classification for words that can have different tags depending on their context. We used the NLPNET POS tagger instead [27], while the input is the trained Word Embeddings of the annotated corpus.

3) Document Categorization (DC)

The aim of Document categorization is to assign one or more labels to a document. This is useful for environments that deals with a lot of content such as libraries, and publishing companies. Moreover, if this categorization was done on a paragraph or a sentence level, it could be used in a wide variety of other NLP tasks such as sentiment analysis and topic detection. For our application, the goal is to assign labels to several articles, these labels present six different categories namely: Culture, Economy, International, Local, Religion and Sports. We used Kalimat Multipurpose Arabic Corpus that contains 20291 articles, collected from an Omani newspaper, divided into the 6 categories mentioned above [26]. Table 9 presents the statistics of the training dataset with an example of a simple input. We followed the same architecture we used for NER tagging, the only difference is that the input for NER was; Single tokens as word representations, and their NER tags, while for DC the input is a sequence of vectors that represents the articles from the DC training data, divided into 6 categories.

Total of articles	Culture	Economy	International	Local	Religion	Sports
20291	2782	3468	2035	3596	3860	4550
Example of an input	تتواصل الاجتماعات التعريفية للمشروع الوطني لحصر الصناعات الحرفية والحرفيين في مختلف … $[[-0.02240.0120036, \ldots, 0.0508491], \ldots, [-0.00786473, \ldots, 0.0131017]]$					

Table 9. Statistics of the DC annotated data

4) Sentiment Analysis (SA)

Sentiment Analysis is the task of detecting people's opinions, sentiments, evaluations and attitudes, towards entities such as products, services, issues, and topics. It is an important NLP research field, and its application is visible in a variety of domains such as politics, commerce, education and health. For our experiment, we perform the SA on a sentence level, where the objective is to identify if a certain sentence holds an opinion and the orientation of it. The annotated training data we used was a fusion of ArsenTD that holds many annotated Arabic blogs from [21] and a Twitter dataset for Arabic sentiment analysis provided by the authors of [28]. Table 10 illustrates the statistics of the resulted SA annotated corpus. And once again, we followed the same architecture we used for NER tagging and DC, while the input is the pair: a sequence of vectors representing the annotated sentences, and their tags; neutral, positive or negative. We choose the SA application because of its morphological nature. More specifically, to investigate if combining the Arabic dialects in one corpus has an impact of detecting the morphological properties of AD words, we conducted this experiment on the three datasets and a fusion of the Facebook and twitter datasets.

Total of inputs	Positive	Negative	Neutral
6166	2238	2931	997

Table 10. Statistics of the SA annotated data

IV. Results & discussion

The results of our experiments and applications will be discussed on three levels: 1) The effect of the hyper-parameters (i.e. the contextual window and the vector dimension) on the Embeddings' quality. 2) The performance of different Word Embeddings' methods on the different NLP tasks. 3) And the outcome of the corpus source on building Embeddings for different Arabic varieties. The results were computed using the accuracy notation for the intrinsic evaluations, and the F1 score for extrinsic evaluations.

A. The intrinsic evaluation results

1) Word Analogy task results

The results of the word analogy task according to the hyper-parameters of the models are illustrated in Figure 2. The first observation of these results is that neither the context nor the vector dimension has a major effect on the word analogy results for the word2vec models (i.e. CBOW & SG). In fact, it is an irregular negligible change of 0.1% to 1.5%, except for the Negative Sampling SKIP-Gram model where the results increase slightly with decreasing the vector dimension (see Figure 2 (b)). However, we observe a significant improvement in the Glove models when we increase the contextual window, this improvement is more noticeable for Twitter and Facebook datasets, where we see a difference of more than 20% (see Figure 2 (n) and (o)). It is important to note that the ratio of MSA content in social media datasets is relatively lesser than the AD content, which results in the rarity of MSA words. Also, the benchmarks that we used in this evaluation are written in MSA. Thus, we can deduce that Glove benefits from the contextual window when it comes to rare words, and, that the common hypothesis of a larger context window emphasizes the learning of domain similarity between words can be applied for Arabic as well.

A second observation is the performance's difference of the models when trained on different data sets; For the word2vec models, Facebook and Twitter have a similar accuracy on sematic and syntactic questions, Wikipedia is instead about 15% more accurate on the syntactic questions (see figure 2 (j), (k) and (l)). We attribute this behaviour to the fact that Wikipedia is encyclopedic and notion-centric which results in syntax-rich vectors that can be beneficial in some practical application such as parsing tasks.

The overall word analogy test result is the mean of the correct syntactic and semantic answers. To have a clear comparison between all the models on this task, we selected the best result achieved by each model. Table 11 summarize these results. Glove was the best model for both Arabic variant (i.e. MSA & AD) with an accuracy of 47.2% for Wikipedia, and 44% for the Facebook dataset. All the trained models performed better than the previous pre-trained FastText vectors on Wikipedia dataset presented in [8] which has an accuracy of 26.93 with CBOW architecture and 17.14% with the Skip-Gram architecture.

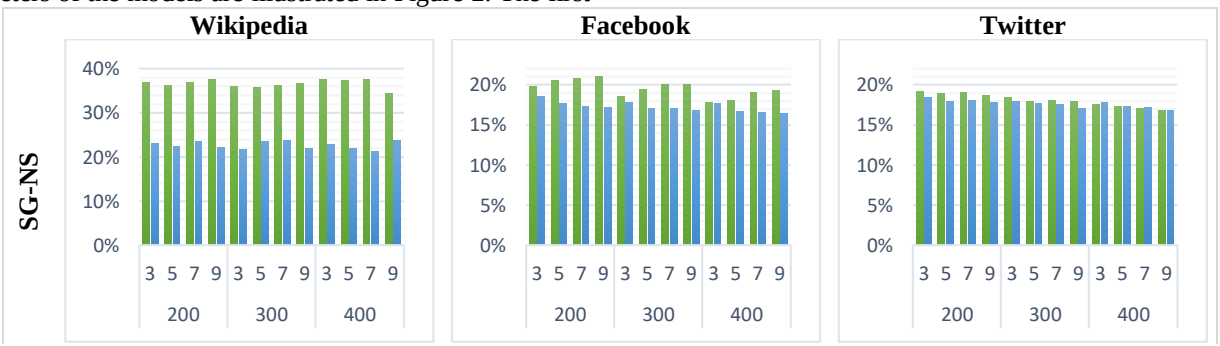


Figure 2. Accuracy results according to the Word Analogy Task. The green bars indicate the syntactic task, whereas the blue ones the semantic. The first horizontal axis presents the contextual window, the second one presents the vector's dimension.

	Wikipedia			**Facebook**			**Twitter**		
	Acc	**Dim**	**W**	**Acc**	**Dim**	**W**	**Acc**	**Dim**	**W**
SG-NS	30.2%	400	3	18.8%	200	3	19.2%	200	3
SG-HS	30.6%	200	3	19.6%	200	3	20.0%	300	3
CBOW-NS	33.0%	200	3	19.2%	300	5	19.9%	200	3
CBOW-HS	33.3%	200	9	19.7%	400	9	20.3%	400	9
Glove	**47.2%**	400	9	**44.0%**	400	9	**46.4%**	400	9
Fast-Text SG	17.1%	300	10						
Fast-Text CBOW	26.9%	300	10						

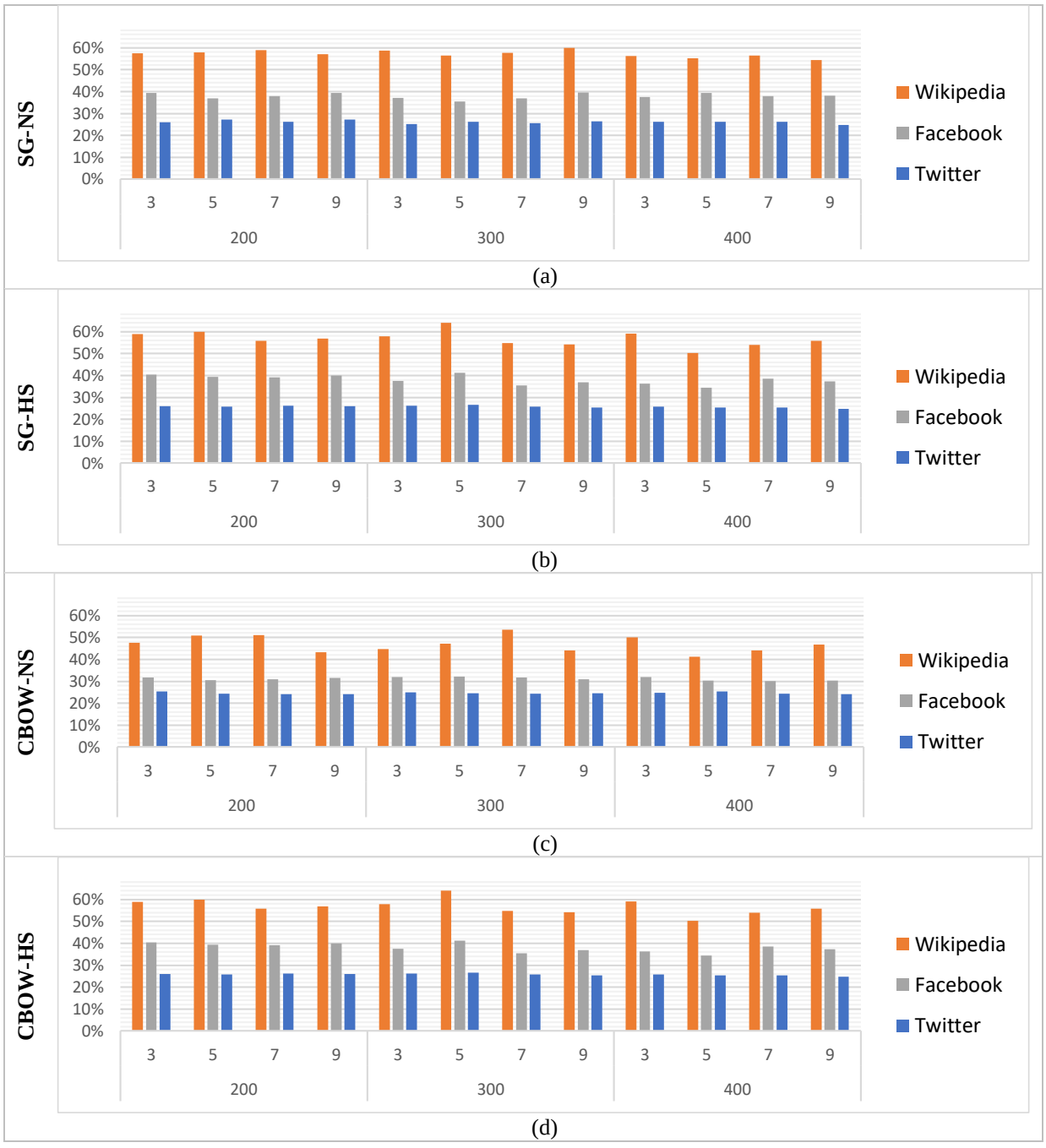
Table 11. Summarization of the best model's evaluation results according to the word analogy test. The best result for each dataset is highlighted in bold.

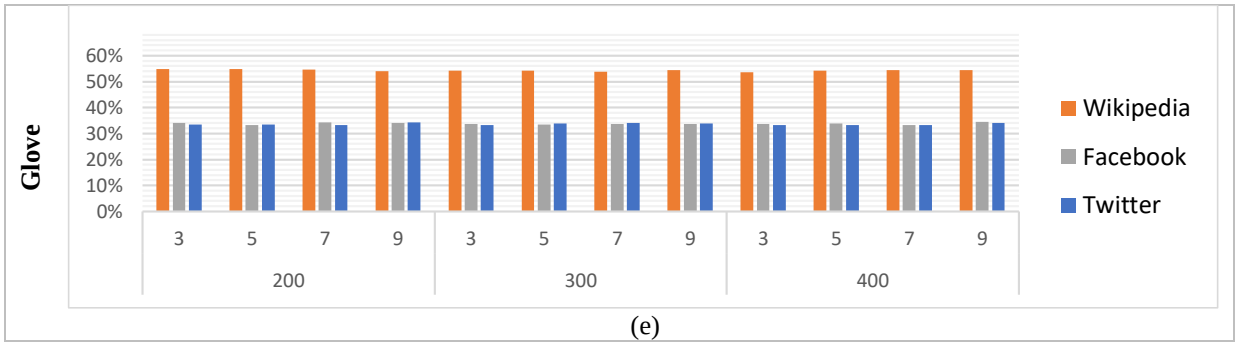
2) Concept categorization results

Figure 3 Summarizes the concept categorization results of each hyper-parameter combination for the different Word Embeddings' models trained on different datasets. The results obtained from the word2vec models are not as affected with the vector's dimension as they are affected by the contextual window; we notice a modest improvement of 4% to 6 % for each dimension by fine-tuning the word context length (see figure 3 (b)). We can speculate that, for word2vec models, the context affects the accuracy level of the relation recovery tasks. This observation, however, does not apply on Glove models, where we see no considerable change by fine-tuning both hyper-parameters (see figure 3 (e)).

Comparing between the two social media datasets, the results of the word2vec models trained on Twitter are much worse than the Facebook ones (see figure 3 (d)). This is probably due to the small vocabulary size and the ratio of MSA/AD content in our twitter dataset.

Table 12 shows the best concept categorization results obtained from the different models for each dataset. For Wikipedia and Facebook datasets, the skip-gram HS model with a vector dimension of 300 and a window length of 7 achieved 64.1% and 41.2% as best results, followed by the skip-gram NS, Glove, then CBOW. Repeatedly, all the trained models performed better than the FastText vectors on Wikipedia dataset presented in [8]. For the Twitter dataset, Glove had the best result with 34.3 %, followed by Skip-Gram then CBOW. It is worth mentioning that, even if the Glove both hyper-parameters didn't have a major impact on the results, the best ones were obtained with the largest contextual window.





(e)

Figure 3. Accuracy results according to the Concept categorization test. The first horizontal axis presents the contextual window, the second one presents the vector's dimension.

	Wikipedia			Facebook			Twitter		
	Acc	**Dim**	**W**	**Acc**	**Dim**	**W**	**Acc**	**Dim**	**W**
SG-NS	59.8%	300	9	39.5%	300	9	27.2%	200	9
SG-HS	**64.1%**	300	5	**41.2%**	300	5	26.6%	300	5
CBOW-NS	53.5%	300	7	32.1%	300	5	25.4%	400	5
CBOW-HS	54.1%	200	7	33.9%	400	3	25.8%	300	3
Glove	54.8%	400	9	34.5%	400	9	**34.3%**	200	9
Fast-Text SG	36.9%	300	10						
Fast-Text CBOW	40.3%	300	10						

Table 12. Summarization of the best model's evaluation results according to the concept categorization test. The best result for each dataset is highlighted in bold.

B. The extrinsic evaluation results

Figure 4 illustrates F1 measures for the trained models according to three NLP tasks applications (i.e. NER, POS and DC). Roughly speaking, the same observation noted in previous evaluations could be adapted on the extrinsic evaluations as well; the models' contextual window or the vector' dimension doesn't have a consistent regular effect on the results. However, we notice some positive and negative changes while tuning both hyper-parameters. For instance, the Skip-Gram model with the HS architecture, a context window of 3 and a vector dimension of 400 behaved differently on the various datasets in the DC task; Facebook and Twitter performed well while Wikipedia did not (see figure 4 (d)-(f)). Although, when the dimension is altered to 300 Wikipedia had relatively good results. The divergence we notice while fine-tuning the parameters is significant (i.e. an improvement of 12% in some cases, see Figure 4 (k)). Nevertheless, in most cases, these changes are inconsistent and random throw out the different applications and datasets. Another important observation is, on the contrary to previous evaluations, the average results of each model are similar throw out the different datasets.

Figure 5 shows the obtained results on the SA application. The fine-tuning of the contextual window has a noticeable impact on the F1 measure for all the models. In fact, for the Glove models, increasing the context improves accuracy. As for the vectors' dimension, there is a slight and inconsistent impact on the F1 measure. Comparing between the Word Embeddings methods, we notice that the performance of Word2Vec models exceeded the Glove's, and the NS architecture enhanced the results for both the Skip-Gram and CBOW models. In addition to the trained models used in the other applications, for SA, we opted to train additional models using a fusion of Twitter and Facebook datasets. The obtained results proved that, for the Word2Vec models, the new dataset (social media) performed better than all the previews ones with a difference of 8% for some cases (see Figure 5 (a)). This is because the dataset contains MSA and all the varieties of the AD. And on contrary to Wikipedia that has more neutral and encyclopedic outtake on things, social media datasets, in general, contains subjects with an opinionated nature.

Table 13 summarize the best results obtained by each model according to the various NLP applications. For the NER task, on average, the CBOW and Glove models have the best results on the different datasets, followed closely by the Skip-Gram models. For the POS task, the overall performance of Glove is significantly worse than the Word2vec models; the CBOW both architectures have the best results with 90.0% for Wikipedia, 83.4% for Facebook, and 87.3% for Twitter. For the DC application, the overall performance of the different models was good, the Glove and CBOW with the HS architecture had the best results with an F1 score of 95.6% for Wikipedia dataset. As for the SA application, the CBOW-NS trained on Social-media dataset generally outperformed the other models with an F1 score of 70.9%, followed by SG both architecture than Glove.

Figure 4. F1 measures of the extrinsic applications. The green bars present the NER results, the blue bars the POS results, and the yellow bars the DC results. The first horizontal axis presents the contextual window and the second one presents the vector's dimension.

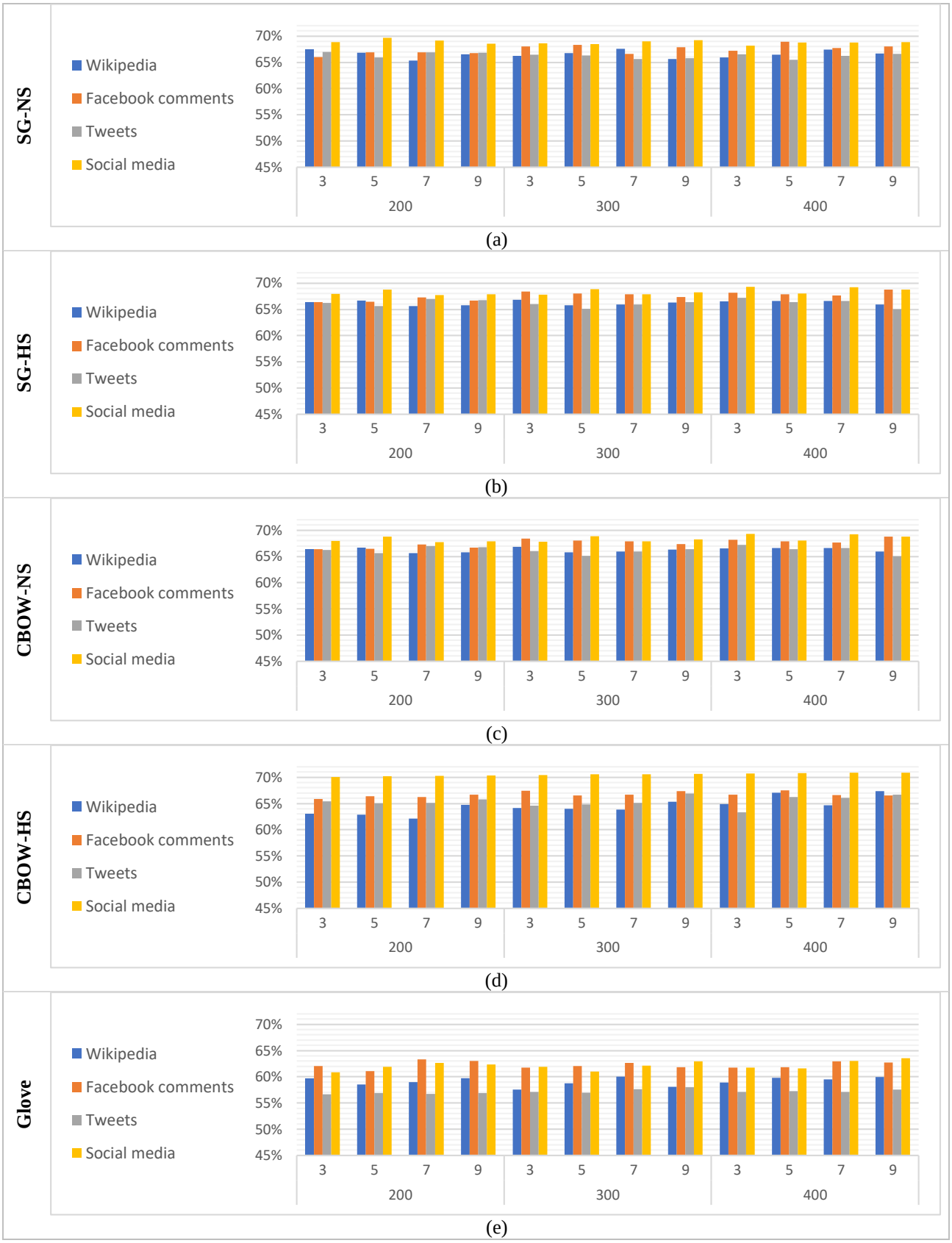


Figure 5. F1 measures of the evaluated models with different hyper-parameters and datasets according to the SA application.

		Wikipedia			Facebook			Twitter			Social media		
		Acc	**Dim**	**W**	**Acc**	**Dim**	**W**	**Acc**	**Dim**	**W**	**Acc**	**Dim**	**W**
NER	**SG-NS**	87.5	400	5	87.8	300	9	**88.0**	200	9			
	SG-HS	87.6	200	5	87.8	400	9	87.7	200	9			
	CBOW-NS	87.8	400	7	87.7	300	3	**88.0**	400	5			
	CBOW-HS	**87.9**	200	7	87.5	200	7	87.6	300	9			
	Glove	**87.9**	200	5	**87.9**	400	3	87.7	200	5			
	Fast-Text SG	87.4	300	10									
	Fast-Text CBOW	86.8	300	10									
POS	**SG-NS**	89.7	400	3	82.5	400	3	85.8	400	3			
	SG-HS	88.7	400	3	82.9	400	3	85.5	400	3			
	CBOW-NS	**90.0**	400	7	81.4	400	3	86.2	400	3			
	CBOW-HS	89.8	400	9	**83.4**	400	3	**87.3**	400	3			
	Glove	77.8	400	3	66.0	400	7	71.0	400	9			
	Fast-Text SG	87.6	300	10									
	Fast-Text CBOW	83.7	300	10									
DC	**SG-NS**	92.2	200	9	93.3	200	5	94.4	300	3			
	SG-HS	**95.6**	400	9	94.4	200	7	**97.8**	400	3			
	CBOW-NS	92.2	400	9	93.3	400	3	92.2	400	5			
	CBOW-HS	95.3	200	5	**96.5**	200	3	91.1	300	9			
	Glove	95.4	200	9	94.4	300	7	94.4	200	5			
	Fast-Text SG	90.0	300	10									
	Fast-Text CBOW	92.2	300	10									
SA	**SG-NS**	67.6	300	7	**69.0**	400	5	67.0	200	3	69.7	200	5
	SG-HS	66.9	300	3	68.8	400	9	67.2	400	3	69.3	400	3
	CBOW-NS	65.9	400	3	65.7	300	3	**67.3**	300	7	69.4	400	9
	CBOW-HS	67.3	400	9	67.5	400	5	66.9	300	9	**70.9**	400	9
	Glove	60.0	300	7	63.3	200	7	58.0	300	9	63.5	400	9
	Fast-Text SG	**69.5**	300	10									
	Fast-Text CBOW	66.4	300	10									

Table 13. Summarization of the best model's F1 measures (%) for each NLP application. The best result for each dataset and application is highlighted in bold.

C. Discussion

Based on the results discussed in the previous section, we deduce that each Word Embeddings' method has its advantages and disadvantages for a specific Arabic NLP application. For applications with a semantic nature such as DC and SA, models created using Word2Vec methods performed better than the ones created using Glove. However, for applications with a syntactic and contextual nature, they performed coequally, with Glove outperforming Word2Vec models in some cases as illustrated in Figure 6 This leads to the conclusion that Word2Vec models are good in capturing the meanings of words, while Glove is better when it comes to the Global information of a context.

The ratio MSA/AD content in social media sources is small, which results in the rarity of MSA words in these datasets. According to the results obtained in the previous section, Glove and CBOW performed well on Facebook and twitter datasets. This proves that these two methods create good presentations for rare words. They are designed to predict the most probable word, and they benefit from a large context to generate presentations for rare words. On the other hand, when the training dataset is relatively small, the SG is better.

FastText models had an overall of poor results in comparison to the other methods, however, for SA application it obtained the best F1 score for Wikipedia dataset. This can be explained by the fact the FastText that we used in this study were trained on different datasets.

FastText is known for generating word presentations for words that don't exist in the training dataset. And, since Wikipedia contains only the MSA variety, and the training benchmarks were a mix of MSA and AD, we deduce that; this method can be used as a solution of the lack of training resources, and can generate decent word presentation for inexistent Arabic words.

The majority of the best results were obtained by models that have the largest dimension. Nevertheless, both hyperparameters behaved differently for the various Arabic NLP applications. For Example, on the concept categorization, the Word2Vec models' context affected the accuracy level but didn't have a major impact on the Word Analogy task. Glove models behaved similarly; in the Word Analogy task, the accuracy increased while increasing the contextual window, although there is an irregular negligible change for other applications. From this discussion, it is obvious that, even if it is inconsistent and random, the tuning of both hyperparameters have a major impact on the accuracy of the different NLP applications.

The source of the training dataset is significant to the accuracy of an Arabic NLP application as well. For instance, Wikipedia and other encyclopedic sources have a motion-centric nature, thus, regardless of the used Word Embeddings' method, the resulted model will be syntax-rich with a lack of notions. Consequently, they will perform well on contextual applications such as; POS and NER, but not as well on applications like DC and SA, in which, the dataset from Social media and blogs sources seem to have better results.

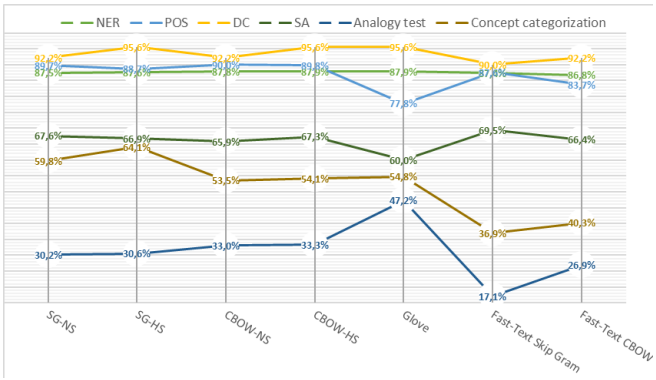


Figure 6. A comparison of the trained models' performance on the Intrinsic and extrinsic applications for Wikipedia dataset.

V. Conclusion & Future works

In this paper, we have tested three word-Embeddings methods (i.e. SG, CBOW and Glove), trained them on three other different datasets composing the different Arabic varieties, and, investigated the quality of the trained models using 12 combinations of the hyper-parameters. We used different intrinsic and extrinsic evaluations methods. In conclusion, our study raises two outcomes; The different stylistic properties of the three datasets and the tuned hyper-parameters had an impact on the semantic and syntactic properties of the generated word representations, subsequently, an impact on the NLP tasks.

Several promising directions remain to be explored; we would like to experiment with other uprising language models, namely, Elmo [29], Bert [30] and XLNET [31], and other classification and clustering training algorithms to get the best results on the NLP applications.

Acknowledgement

This work has been conducted during a collaboration of the first author with *Khaos Research* Labs Spain. All the experiments in this paper have been conducted on the Labs' multiprocessor at Malaga University.

References

[1] S. S. Neeraj, B. Yoshua, D. R ́ejean, V. Pascal, and J. Christian, "A Neural Probabilistic Language Model," *J. ofMachine Learn. Res.*, vol. 3, pp. 1137–1155, 2003.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *IJCAI Int. Jt. Conf. Artif. Intell.*, pp. 4069–4076, 2013.

[3] N. Hartmann, E. Fonseca, C. Shulby, M. Treviso, J. Rodrigues, and S. Aluisio, "Portuguese Word Embeddings: Evaluating on Word Analogies and Natural Language Tasks," 2017.

[4] R. Tripodi and S. L. Pira, "Analysis of Italian Word Embeddings," 2017.

[5] M. S. Zahedi, M. H. Bokaei, F. Shoeleh, M. M. Yadollahi, E. Doostmohammadi, and M. Farhoodi, "Persian Word Embedding Evaluation Benchmarks," in *26th Iranian Conference on Electrical Engineering, ICEE 2018*, 2018, pp. 1583–1588.

[6] A. B. Soliman, K. Eissa, and R. E.-B. Samhaa, "AraVec : A set of Arabic Word Embedding Models for use in Arabic NLP," in *Procedia Computer Science.*, 2017, no. September, pp. 256–265.

[7] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations ofWords and Phrases and their Compositionality," in *NIPS'13 Proceedings of the 26th International Conference on Neural Information Processing Systems*, 2013, vol. 2, pp. 3111–3119.

[8] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[9] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," 2017.

[10] A. Balahur, M. Kabadjov, J. Steinberger, R. Steinberger, and A. Montoyo, "Challenges and solutions in the opinion summarization of user-generated content," *J. Intell. Inf. Syst.*, vol. 39, pp. 375–398, 2012.

[11] M. Javed, M. M. A. Baig, and S. A. Qazi, "Unsupervised Phonetic Segmentation of Classical Arabic Speech Using Forward and Inverse Characteristics of the Vocal Tract," *Arab. J. Sci. Eng.*, vol. 45, no. 3, pp. 1581–1597, 2020.

[12] F. S. S. Alotaibi, "Fine-grained Arabic Named Entity Recognition," UNIVERSITY OF BIRMINGHAM, 2015.

[13] A. Shoufan and S. Al-Ameri, "Natural Language Processing for Dialectical Arabic: A Survey," pp. 36–48, 2015.

[14] R. Al-sabbagh and R. Girju, "YADAC : Yet another Dialectal Arabic Corpus," in *Lrec 2012 - Eighth International Conference on Language Resources and Evaluation*, 2012, pp. 2882–2889.

[15] Twitter, "About Twitter's APIs," *Twitter*, 2019. [Online]. Available: https://help.twitter.com/en/rules-and-policies/twitter-api. [Accessed: 11-Oct-2020].

[16] C. AZROUMAHLI, Y. El Younoussi, O. Moussaoui, and Y. Zahidi, "An Arabic Dialects Dictionary Using Word Embeddings," *Int. J. Rough Sets Data Anal.*, vol. 6, no. 3, pp. 18–31, 2019.

[17] M. A. ELAffendi, I. Abuhaimed, and K. AlRajhi, "A simple Galois Power-of-Two real time embedding scheme for performing Arabic morphology deep learning tasks," *Egypt. Informatics J.*, 2020.

[18] S. Alotaibi and M. B. Khan, "Sentiment Analysis Challenges of Informal Arabic Language," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 2, pp. 278–284, 2017.

[19] NLTK, "Natural Language Toolkit," *Apache*, 2015. [Online]. Available: https://www.nltk.org/. [Accessed: 12-Oct-2020].

[20] M. T. Alrefaie, "arabic stop words," *Github*, 2017. [Online]. Available: https://github.com/mohataher/arabic-stop-words/blob/master/README.md. [Accessed: 11-Oct-2020].

[21] R. Baly and K. B. S. Alaa Khaddaj, Hazem Hajj, Wassim El-Hajj, "ArSentD-LEV : A Multi-Topic Corpus for Target-based Sentiment Analysis in Arabic Levantine Tweets," *OSACT3*, no. March, pp. 37–43, 2018.

[22] C. Azroumahli, Y. El Younoussi, and F. Achbal, "An

overview of a distributional word representation for an arabic named entity recognition system," in *Advances in Intelligent Systems and Computing*, vol. 737, 2018, pp. 130–140.

[23] B. Mohit, N. Schneider, R. Bhowmick, K. Oflazer, and N. A. Smith, "Recall-oriented learning of named entities in Arabicwikipedia," *EACL 2012 - 13th Conf. Eur. Chapter Assoc. Comput. Linguist. Proc.*, pp. 162–173, 2012.

[24] M. Polignano, M. De Gemmis, P. Basile, and G. Semeraro, "A comparison of Word-Embeddings in Emotion Detection from Text using BiLSTM, CNN and Self-Attention," *ACM UMAP 2019 Adjun. - Adjun. Publ. 27th Conf. User Model. Adapt. Pers.*, no. June, pp. 63–68, 2019.

[25] B. Kaliraman and M. Duhan, "Feature Extraction and Classification of EEG Signals Using Machine Learning Algorithms for Biometric Systems," *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.*, vol. 12, pp. 339–348, 2020.

[26] M. El-haj and R. Koulali, "KALIMAT a Multipurpose Arabic Corpus," *Second Work. Arab. Corpus Linguist.*, pp. 22–25, 2013.

[27] E. R. Fonseca and J. L. G. Rosa, "A two-step convolutional neural network approach for semantic role labeling," *Proc. Int. Jt. Conf. Neural Networks*, no. September 2016, 2013.

[28] N. a Abdulla, N. a Ahmed, M. a Shehab, and M. Al-ayyoub, "Arabic Sentiment Analysis: Lexicon-based and Corpus-based," *Jordan Conf. Appl. Electr. Eng. Comput. Technol.*, vol. 6, no. 12, pp. 1–6, 2013.

[29] M. E. Peters *et al.*, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018, pp. 2227–2237.

[30] Akshay Prakash, "BERT: Bidirectional Encoder Representations from Transformers," *medium*, 2019. [Online]. Available: https://medium.com/swlh/bert-bidirectional-encoder-representations-from-transformers-c1ba3ef5e2f4. [Accessed: 25-Jun-2020].

[31] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," pp. 1–18, 2019.

Author Biographies

AZROUMAHLI Chaimae, received her Master's Degree in Education Mathematics and technologies in 2015. She joined SIGL Laboratory (French Acronym for Systems and Software Engineering) at the National School of Applied Sciences Tetuan, Abdel Malek Essa âdi University, Morocco as a PhD student in 2017. Her research interests are Artificial intelligence, Machine learning, Semantic web technologies and NLP.

Maciej Rybiński, received his PhD in Computer Science from the University of Malaga in 2017. He is currently doing a Postdoc in the LASC group of Data61 at CSIRO, Sydney, Australia. His research is focused on the use of semantics and natural language processing, with knowledge extraction and engineering and semantic search being the most important application areas.

El Younoussi Yacine, is a full professor at the National School of Applied Sciences Tetuan, Abdel Malek Essa âdi University, Morocco. He is the Head of the Department of Computer Science and member of the SIGL Laboratory (French Acronym for Systems and Software Engineering). His research interests are Arabic NLP, data analysis, and Big Data.

Jos é F. Aldana Montes, is a full Professor of the area of Computer Languages and Systems in the Department of Languages and Computer Sciences of the University of Malaga. He is the head of Khaos Research at Malaga University. His research interests are Semantic Middleware, Semantic Web, Semantic Integration of Data and Applications and Extensions of the Databases with Formal Semantics.