IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 37, No. 3, June, 2023, pp. 753-759
Journal homepage: http://iieta.org/journals/ria

Towards Amazigh Word Embedding: Corpus Creation and Word2Vec Models Evaluations

Hassan Faouzi[1*], Maria El-Badaoui[2], Mohammed Boutalline[3], Adil Tannouche [4], Hamid Ouanan[5]

[1] Systems Engineering Laboratory (LGS), Higher School of Technology (EST-FBS), Sultan Moulay Slimane University (USMS), Fkih Ben Salah 23203, Morocco
[2] LASTI Laboratory, National School of Applied Sciences, Sultan Moulay Slimane University (USMS), Khouribga 25000, Morocco
[3] Systems Engineering Laboratory (LGS), National School of Applied Sciences (ENSA-BM), Sultan Moulay Slimane University (USMS), Beni Mellal 23000, Morocco
[4] Laboratory of Engineering and Applied Technology (LITA), Higher School of Technology (EST-BM), Sultan Moulay Slimane University (USMS), Beni Mellal 23000, Morocco
[5] Information Processing and Decision Support Laboratory (TIAD), National School of Applied Sciences (ENSA-BM), Sultan Moulay Slimane University (USMS), Beni Mellal 23000, Morocco

Corresponding Author Email: faouzi.hassan.mi@gmail.com

https://doi.org/10.18280/ria.370324

Received: 13 January 2023
Accepted: 5 April 2023

Keywords:
NLP, word embedding, Word2Vec, CBOW, Skip gram, corpus, Python, Tifinagh

ABSTRACT

Distributed representations of words in a vector space help learning algorithms to model semantic notions of word similarity and distances in sentences. Most of the existing researches have been done on the Latin, Arabic, and other language, while the Amazigh language is ignored. In this paper, we try to build a first model word embeddings for Amazigh language and describe the steps needed to build it. Therefore, we implement a Word2Vec that a combination of two techniques – CBOW (Continuous bag of words) and Skip-gram to transform words written in Tifinagh to vector form. To obtain the highest performance, we evaluate two parameters of Word2Vec include Word2Vec model architecture and vector dimension. This evaluation process was implemented towards our proposed corpus collected on Amazigh websites for different domains. The result shows that the highest accuracy values are obtained under the combination of CBOW model and 300 dimensional vector.

1. INTRODUCTION

Undoubtedly, the abundance of electronic documents available in the internet such as research articles, reports, conference papers and other formats makes text analysis one of the most important and critical issue.

Automatic Natural Language Processing (NLP) is a field at the intersection of machine learning and linguistics and it is a very important topic in information retrieval. Find new techniques of text analysis are one of the main tasks of almost most researchers in academia and industry. This domain is very wide and incorporates tools for analyzing, processing and generating text. This is why we find the NLP behind many applications such as:

- Language translation applications.
- Check and correct grammatical mistakes in the text.
- Interactive Voice Response applications used to respond to certain users' requests. and increases customer satisfaction by automating conversations.

Traditionally, standard word representations treat words as one-hot vectors with the same vocabulary size and most positions are all 0, and only one dimension is 1. This structure leads to high dimensionality of words and it is very difficult to catch complex linguistic properties of words in large corpus. To address this issue, a lot of semantic-based approaches are proposed in literature, but unfortunately these researches are focused on traditional languages, such as English, we also noticed the total absence of research dealing with the Amazigh language, because the cultural heritage of this language is essentially oral and the vast majority of the corpus already transcribed in Latin. However, Amazigh and English have different writing rules and syntax, so it is necessary to know the implementation of these algorithms in the specific domain language, i.e., Amazigh language. In addition, it is a very complicated task to adopt an embedding model (of any language) directly for the limited resource languages such as Amazigh due to the scarcity of resources.

In this article, we use Word2vec [1], proposed and supported by Google to process the text data written in Tifinagh. This encoding technique is not a single algorithm, but it includes two learning models; Continuous Bag of Words (CBOW) and Skip-gram [2, 3]. Skip-gram predicts the context given a word and CBOW predicts the word given its context.

The Word2Vec has different settings, the most important are:

- The dimensionality (N) of the vector space to be constructed, i.e., the number of numerical descriptors used to describe the words. So, each word represents a point in D dimensionality space and synonyms words are points closer to each other.
- The size of the context of a word, i.e., the number of terms surrounding the word in question.
- The initial learning rate.
- The number of iterations (epochs) over the corpus.

— Number of worker threads to train the model (=faster training with multicore machines).
— Training algorithm: skip-gram or CBOW.
— Is hierarchical softmax will be used for model training or negative sampling will be used.

This Word2Vec uses a neural network, and generates the distributed representation that corresponds to the weights between the input layer and the hidden layer on the network. Though it also generates the weights between the hidden layer and the output layer together with this distributed representation, the output embedding is not generally used.

So, we developed the proposed Word2Vec model for the Amazigh language to address the challenges presented by the lack of large data, the complexity of the Tifinagh characters, and the unique linguistic features of the language. By representing words in a numerical vector space, the model enables more efficient and effective natural language processing tasks such as text classification, sentiment analysis, and language translation. Word embeddings capture the semantic and syntactic relationships between words, facilitating machine understanding and manipulation of natural language. As a result, the proposed model has significant real-world applications, such as improving the accuracy of machine translation and speech recognition systems for Amazigh. However, there are also challenges and opportunities in implementing the model, such as the lack of high-quality training data and the need to adapt the model to the unique linguistic features of the language.

This paper is organized as follows: Section 2 provides a brief overview of related work in the field of using Word2Vec models; Section 3 presents the proposed system for using Word2Vec for Amazigh language written in Tifinagh. The key contribution of this section is the development of a new model that is specifically designed for the unique characteristics of the Amazigh language, including the use of pre-processing techniques and adjustments to the Word2Vec algorithm's hyperparameters; in section 4, experimental results and discussion are reported; Section 5 concludes the paper and presents the main findings of the study, including the development and evaluation of two word embedding models for the Amazigh language. It introduces future work that aims to improve model accuracy through the use of a larger corpus, as well as exploring character level embedding and applying models to Amazigh sentiment analysis and named entity recognition.

2. RELATED WORK

Word embedding refers to a set of learning methods in order to represent words by vectors of real numbers. It is an important topic in natural language processing (NLP) [2-11]. Distributed word representation can better reveal the semantic information of natural language words. This is because it doesn't use one-hot representations, unlike other NLP tasks. For example, it's been used to model language [12] as well as recognize named entities [13] in texts. It's also been used to classify text [14], answer questions [15].

One of the major advances distributed word in the field of NLP is word2vec proposed by Mikolov et al. in 2013 [1, 16]. This technique is based on deep learning, which used very successfully in many areas [17-21], with two layers and seeks to learn the vector representations of the words composing a text, so that the words which share similar contexts are represented by close digital vectors. Word2vec predicts words based on context with two different neural models: Continuous Bag of Words (CBOW) and Skip-Gram.

2.1 CBOW

This architecture makes it possible to predict a word according to its context, which matches previous words and subsequent words. In this architecture, the projection layer is shared by all the words: all the words are projected in the same position. So, learning word embedding consists of predicting a word based on its context. This is done by calculating the sum of the word embedding of the context, then by applying to the resulting vector a log-linear classifier to predict the target word. Finally, the model compares its prediction with reality and corrects the vector representation of the word by the back propagation of the error gradient. In fact, CBOW tries to maximize the following Eq. (1) [22]:

$$\frac{1}{|\nu|}\sum_{t=1}^{|\nu|}\log\left[p\left(W_t \middle/ W_{t-c},\ldots,W_{t-2},W_{t-1},W_{t+1},W_{t+2},\ldots,W_{t+c},\right)\right] \quad (1)$$

With $\{W_{t-c},\cdots, W_{t+c}\}$ sequence of training words.

The implementation of CBOW architecture will focus on five parts:

— Build the corpus vocabulary
— Build a CBOW (context, target) generator
— Build the CBOW model architecture
— Train the Model
— Get word embedding

2.2 Skip-gram

The Skip-gram model has the comparable architecture as CBOW. It reverses the input and the output of the neural network used by the CBOW model [16]. So, if the previous model made the word prediction using surrounding words (context), the skip-gram model plays the role of predicting the context of a given target word by maximizing the logarithmic probability.

Given a set of training words $\{W_{t-c},\cdots, W_{t+c}\}$, the model maximizes the following average log probability to predict the context of the current target word Eq. (2):

$$\frac{1}{|\nu|}\sum_{t=1}^{|\nu|}\sum_{j=t-c,\,j\neq t}^{t+c}\log\left[P\left(w_j \middle| w_t\right)\right] \quad (2)$$

where $|\nu|$ is the number of words in the corpus and c is the size of the dynamic context of W_t.

The implementation of this architecture is based also on five steps:

— Build the corpus vocabulary.
— Build a skip-gram [(target, context), relevancy] generator.
— Build the skip-gram model architecture.
— Train the Model.
— Get word embeddings.

2.3 CBOW vs skip gram

In the CBOW model, the distributed representations of context (or surrounding words) are combined to predict the word in the middle. While in the Skip-gram model, the distributed representation of the input word is used to predict the context. In fact, CBOW tends to find the probability of a word occurring in a context. So, it generalizes over all the different contexts in which a word can be used. While Skip gram tends to study different contexts separately.

In our model we use Negative sampling technique to improve the efficiency of the Word2Vec model training by reducing the number of times that the model has to update the weights of all the neurons in the output layer. This is achieved by randomly selecting a small subset of "negative" words that are unlikely to be found near the target word, and updating their weights as well, instead of updating the weights of all the other words in the vocabulary. Because in the case of the Amazigh language written in Tifinagh, the use of negative sampling was particularly important due to the limited resources of the available training corpus. Training the model using the standard approach of updating the weights of all words in the vocabulary at each iteration would have been computationally expensive and time-consuming. By using negative sampling, our model be able to reduce the number of times that the model had to update the weights of all the neurons in the output layer, while still achieving accurate word embeddings for the language. This not only improved the efficiency of the model training but also we allowed to experiment with different hyperparameters and settings to optimize the model's performance.

3. METHODOLOGY

Proposed working model is based on the NLP approach that extracts and captures the semantic and syntactic information of words written with Tifinagh characters. This model helps to:

- Suggest similar words to the word being subjected to our prediction model.
- Group words of similar characteristic together and dissimilar far away.
- Convert the higher dimensional representation of the text into lower dimensional of vectors.

To evaluate our module we had to collect the necessary dataset written with Tifinagh given the lack of public resources to use it in the field of NLP. So below is the step-by-step method to implement our model using Word2vec.

3.1 Construction of dataset

The first step in implementing a machine learning model or implementing natural language processing is data collection (corpus).

A corpus is a collection of real text written or dictated by native speakers of the language or dialect. It can contain anything from newspapers, novels, books, movies and tweets. In natural language processing, corpus contain text and language data that can be used to train machine learning systems.

Corpus Amazigh is a limited resource language, and it is not rich on the social media websites. So, in collecting data for our Word2Vec model for the Amazigh language written in Tifinagh, we relied on various sources, including online resources and manual documents. One of the primary sources we used was the websites like the link of Royal Institute of Amazigh Culture (IRCAM) in Morocco [23], which provides a wealth of information on the Amazigh language and culture. We also consulted manual documents, such as the "Manual of Amazigh Language for Students" (Tizlatin Inu) and the "Manual of Royal Institute of Amazigh Culture" (Ilagan), which provided syntax and vocabulary of the language. Additionally, we worked with native speakers and language experts to ensure the accuracy and quality of the data collected. By using a combination of online and manual resources, we were able to gather a diverse range of data to train our Word2Vec model and generate accurate word embeddings for the Amazigh language written in Tifinagh. During this collection, we have included sentences from multiple domains such as general news, literature, sports, politics, and culture.

3.2 Text preprocessing and cleaning

Text preprocessing and cleaning is crucial step in NLP text mining. This phase includes removing stop words and punctuation. However, even that stop words are many in the dataset, they should not appear as vocabulary words during the modeling phase. So, stop words are removed during the preprocessing phase.

Unfortunately we cannot use the library nltk.corpus from NL Toolkit to get the Amazigh language stop word list, because it not yet integrated in this Toolkit. So, we had to build the list of stop words manually based on grammar rules of this language. We include all stop words such as:

‘ⵏ’,’ⵉ’,ⵙ’,’ⴳ’,’ⴷⵉ’,’ⵣⴳ’,‘ⵖⵔ’,’ⴷⴰⵔ’,’ⵅⵀ’,’ⴰⴳⴷ’,’ⴷ’,’ⴳ
ⵔ’

Thus, we must remove these from our dataset because they have low predictive power.

For example, consider this sentence:

ⵜⴳⴰ ⵜⵖⵔⵎⵜ ⵏ ⵀⴰⵙ ⵙⴳ ⵜⵖⵔⵎⵉⵏ ⵏ ⵣⵉⴽⴽ ⴳ ⵍⵎⵖⵔⵉⴱ.

After removing stops words:

✍ ⵏ
✍ ⵙⴳ
✍ ⴳ

the sentence becomes:

ⵜⴳⴰ ⵜⵖⵔⵎⵜ ⵀⴰⵙ ⵜⵖⵔⵎⵉⵏ ⵣⵉⴽⴽ ⵍⵎⵖⵔⵉⴱ.

The example below shows how we had prepared the corpus for building our model using Python language:

```
import re
def clean_text (
string: str,
punctuations=r'''!()-[]{};:'"\,<>./?@#$%^&*_~''',
stop_words=['ⵏ','ⵉ',ⵙ','ⴳ','ⴷⵉ','ⵅⴳ','ⵖⵔ','ⴷⴰⵔ','ⵅⵎ','ⴰⴳⴷ','ⴷ','ⴳⵔ'…..]) -> str:
# Removing the urls from text
string = re.sub(r'https?://\S+|www\.\S+', '', string)
# Cleaning the html tags
string = re.sub(r'<.*?>', '', string)
# Removing the punctuations
for x in string:
if x in punctuations:
string = string.replace(x, "")
# Removing stop words the Amazigh language
string = ' '.join([word for word in string.split() if word not in stop_words])
# Removing the whitespaces
string = re.sub(r'\s+', ' ', string).strip()
return string
```

3.3 Tokenization

Tokenization is the task of splitting a sequence of text into units called token throwing away certain characters, such as punctuation. The objective of this process is to interpret the meaning of the text by analyzing the sequence of words. We can tokenize text at a variety of units including: characters, words, sentences, lines, paragraphs, etc. In our model we used words units, so the input sentence is broken apart every time a white-space is encountered. Here is an example of tokenization implemented in our model (Figure 1):

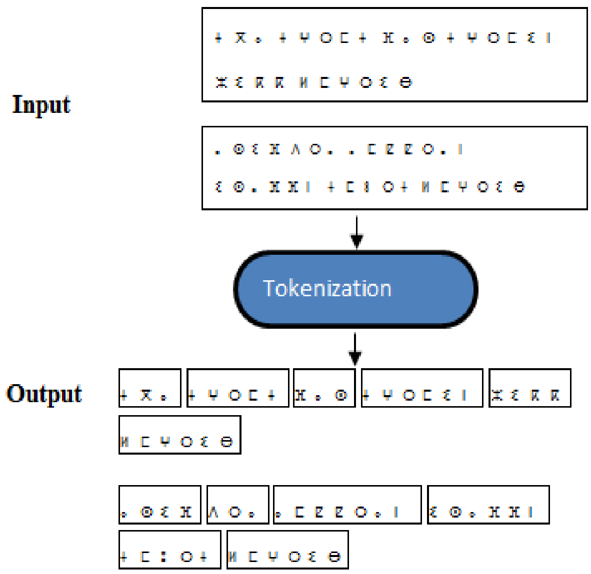

Figure 1. Splitting text into token throwing away

The word_tokenize() function is a popular tokenization technique due to its simplicity and efficiency in breaking down a text into individual words. It uses a set of pre-defined rules to split text into individual words based on whitespace and punctuation, making it highly effective at handling common scenarios such as contractions, hyphenated words, and abbreviations without requiring any additional configuration. Additionally, it is computationally efficient, making it suitable for processing large volumes of text and highly customizable. for all of these advantages, we split sentences using the word_tokenize() function in our implementation. The output of this function provided as input for further text processing. The code below explains how to use the word_tokenize() function:

```
from nltk.tokenize import word_tokenize
for sentence in sentences:
words = word_tokenize(sentence)
print(words)
print()
```

After this step the large quantity of our corpus is divided into smaller parts called tokens. These tokens help in understanding the context of Amazigh words and interpret the meaning of the text by analyzing the sequence of the words.

3.4 Applying Word2vec

3.4.1 Skip – gram

The skip gram algorithm tries to use the current word to predict its neighbors (its context) as shown in Figure 2. For this reason, we will randomly initialize the vector for each word in our corpus. After that, we will go through each position p and we will define the center word at that position and a window of size m in order to select the context words.

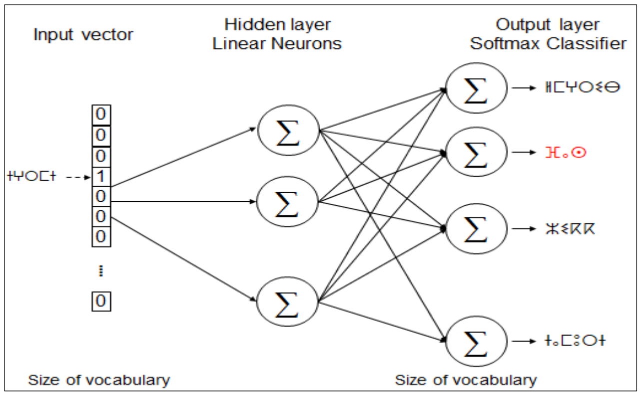

Figure 2. Skip gram

The variables we'll be using in Skip gram are:

- ✓ The dictionary of unique words present in our corpus. This dictionary is known as vocabulary and is known words to the system. Vocabulary is represented by vector 'V'.
- ✓ N is the number of neurons present in the hidden layer of neural network.
- ✓ The window size is the maximum context location at which the words need to be predicted. The window size is denoted by c. For example, in our given architecture (Figure 2) the window size is 2, therefore, we will be predicting the words at location (Wp -2), (Wp-1), (Wp +1) and (Wp +2).
- ✓ The context window is the number of words to be predicted that can occur within a given word range. The value of the context window is twice the window size, ie 2*c, denoted by k. For our given architecture (Figure 2) the value of the context window is 4.
- ✓ The dimension of an input vector is equal to |V|. Each word is encoded using one hot encoding.
- ✓ The weight matrix for the hidden layer (W) has dimension [|V|, N].
- ✓ The output vector of the hidden layer is H[N].

- ✓ The weight matrix between the hidden and the output layer (W') is of dimension [N, |V|].
- ✓ The dot product between W' and H gives us an output vector U[|v|].

3.4.2 CBOW

The CBOW model architecture tries to predict the current target word based on the source context words. Considering a simple sentence, "ⵜⵅⴰ ⵜⵖⵔⵎⵜ ⵊⴰⵙ ⵜⵖⵔⵎⵉⵏ ⵥⵉⴽⴽ ⵍⵎⵖⵔⵉⴱ", this can be pairs of (context_window, target_word). Therefore, if we consider a context window of size 2, we have examples like:

([ⵜⵅⴰ, ⵊⴰⵙ], ⵜⵖⵔⵎⵜ),([ⵜⵖⵔⵎⵜ, ⵜⵖⵔⵎⵉⵏ], ⵊⴰⵙ), ([ⵊⴰⵙ, ⵥⵉⴽⴽ], ⵜⵖⵔⵎⵉⵏ).

Thus, the model attempts to predict the target word based on the context window words as shown in Figure 3 using context window of size 4.

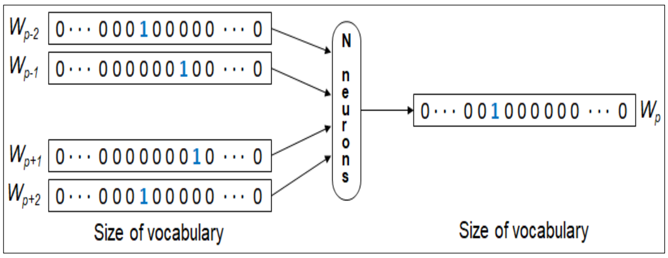

Figure 3. CBOW

4. EXPERIMENTAL RESULT

In order to evaluate the performance of word embedding for Amazigh language, we defined the following workflow for analyzing and evaluating the results produced by the CBOW and Skip gram tools:

a) We trained the corpora using these two techniques of word2vec.

b) Tested different values of dimensionality of vector space (200, 300,400 and 500) and corpus size.

c) Varying the context window size and the number of training epochs.

d) Other parameters were not varied to keep the complexity of results manageable.

The first experiment shows the relationship between word vectors length, corpus size and how they influence the results obtained by the two techniques of word embedding. The outcome of this experiment is summarized in Table 1.

Table 1. Accuracy using different word vectors dimensions and corpus sizes

Number of dimensions	Relative corpus size			
	10%	20%	50%	100%
200 (Skip gram)	13.11%	18.78%	32.77%	76.84%
200 (CBOW)	11.23%	12.66%	28.34%	77.76%
300 (Skip gram)	14.12%	22.26%	37.65%	81.43%
300 (CBOW)	11.36%	14.44%	34.55%	86.45%
400 (Skip gram)	10.33%	21.66%	32.96%	78.43%
400 (CBOW)	10.11%	13.33%	29.15%	79.44%
500 (Skip gram)	9.33%	18.22%	30.34%	74.66%
500 (CBOW)	8.55%	12.98%	26.36%	76.35%

From results we concluded that using the smaller vectors dimensions and law data corpus was not sufficient to encode all word relationships; this is why we did not register a significant difference in the performance of the two techniques, which stays below expectations. Increasing the portion of corpus size and the number of dimensions lead to improvements the performance of the two techniques especially the CBOW model. When the dimensionality was around 300 the results stopped improving, or they even degraded.

In the next experiment two parameters were relevant for CBOW and Skip Gram: The number of training epochs and context window size (we fixed the number of dimension at 300). The detailed results obtained for these two parameters values can be found in Tables 2-4.

Table 2. Accuracy using different context window size and training epochs=30

Model	Context window size (training epochs=30)			
	2	5	7	10
Skip gram	78.07%	79.13%	78.10%	77.41%
CBOW	80.40%	86.13%	81.09%	78.10%

Table 3. Accuracy using different context window size and training epochs=50

Model	Context window size (training epochs=50)			
	2	5	7	10
Skip gram	78.12%	79.15%	78.11%	77.45%
CBOW	80.44%	86.15%	81.12%	78.12%

Table 4. Accuracy using different context window size and training epochs=100

Model	Context window size (training epochs = 100)			
	2	5	7	10
Skip gram	78.33	79.51	78.46	77.63
CBOW	80.67	86.89	81.34	78.15

The Tables 2-4 contain the values of accuracy for different context window sizes for a 30, 50 and 100 numbers of training epochs respectively. From results of our experiments, we notice that the context window size had significant influence on the results when using the CBOW technique and a smaller impact when using the skip gram technique. We also conclude that the number of training epochs does not appear to have a significant effect on the results; fifty training epochs yielded the best results in most experiments.

According to our experiments results we notice the following points:

- CBOW is relatively faster to train data than skip-gram because it uses only one activation function in the output layer of neural networks (softmax).
- CBOW is better for frequently and most common Amazigh words because if a word occurs more often it will have more training words to train.
- Skip-gram is slower but gives better results for the smaller corpus than CBOW.
- We need to optimize the parameters of training data because in the case of 100 (dimension), 2 (window size) and 10000 (words) we have nearly 3 million parameters to train.

5. CONCLUSIONS

In Natural Language Processing, we want computers to understand text like humans. However, to do this, they must translate the text into a vector that computers can use and that they can understand.

Through this paper, we have presented how we have built 2 different word embedding models (CBOW and Skip gram) for the Amazigh language using different resources including online Amazigh newspapers and electronic Journals. We have evaluated these models using the value of the accuracy to demonstrate their ability to detect similarities between words.

The steps of the creation of the word embedding in our model can be summarized in the following points:

- Collect corpus.
- Read the corpus.
- Preprocess corpus.
- Create (x, y) data points.
- Create one hot encoded (X, Y) matrices.
- Train a neural network.
- Extract the weights from the input layer.

In our experiments, the best ranking results were obtained with the CBOW architecture and a vector dimensionality of 300, while worst ranking results were observed with the Skip gram architecture and a vector dimensionality of 500. The results suggest that the CBOW architecture consistently yielded better result ranking than the Skip gram architecture. A higher dimension of the vectors implies a bigger size of the resulting vector model. Therefore, having a vector model with a dimension of 300 requires less memory space and provides better results than a vector model with 400 or 500 vector dimensions.

Comparing the results of this study with similar studies in other languages is challenging, as the linguistic characteristics of each language differ significantly. However, in general, the proposed Word2Vec model for Amazigh language achieves high accuracy in capturing the semantic relationships between words.

While this study has limitations that need to be acknowledged, the proposed Word2Vec model for Amazigh language can still be used in various fields, such as information retrieval, text classification, and sentiment analysis. By capturing the semantic and syntactic information of the language, the model can assist in developing effective natural language processing (NLP) systems for Amazigh speakers, as well as contribute to the development of language learning applications. However, it is important to note that the representativeness of the Amazigh corpus used for training the Word2Vec model may be limited, and the generalizability of the results to other domains or languages may be limited due to the unique linguistic characteristics of each language and the influence of the training corpus on the effectiveness of the model.

Our future work will focus on overcoming the limitations of the proposed Word2Vec model for the Amazigh language. Specifically, we aim to use larger and more diverse corpora to train the model, which will enhance its accuracy and generalizability. Additionally, we plan to investigate the use of character level embeddings to further improve the model's performance. Furthermore, we intend to apply these models to address the challenges of Amazigh sentiment analysis and named entity recognition, which are important tasks in natural language processing.

REFERENCES

[1] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems. https://doi.org/10.48550/arXiv.1310.4546

[2] Brants, T., Popat, A.C., Xu, P., Och, F.J., Dean, J. (2007). Large language models in machine translation. Association for Computational Linguistics, pp. 858-867. https://aclanthology.org/D07-1090.

[3] Collobert, R., Weston, J. (2008, July). A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine Learning, pp. 160-167. https://doi.org/10.1145/1390156.1390177

[4] Duchi, J., Hazan, E., Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12(7): 2121-2159.

[5] Huang, E.H., Socher, R., Manning, C.D., Ng, A.Y. (2012). Improving word representations via global context and multiple word prototypes. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, 1: 873-882. https://aclanthology.org/P12-1092.

[6] Maas, A., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C. (2011). Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 142-150. https://aclanthology.org/P11-1015.

[7] Mikolov, T., Kopecky, J., Burget, L., Glembek, O. (2009). Neural network based language models for highly inflective languages. In 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, Taipei, Taiwan, pp. 4725-4728. https://doi.org/10.1109/ICASSP.2009.4960686

[8] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., Khudanpur, S. (2010). Recurrent neural network based language model. In Interspeech, 2(3): 1045-1048.

[9] TMikolov, T., Kombrink, S., Burget, L., Černocký, J., Khudanpur, S. (2011). Extensions of recurrent neural network language model. In 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Prague, Czech Republic, pp. 5528-5531. https://doi.org/10.1109/ICASSP.2011.5947611

[10] Mnih, A., Hinton, G. (2007). Three new graphical models for statistical language modelling. In Proceedings of the 24th International Conference on Machine Learning, Toronto, Canada, pp. 641-648.

[11] Mnih, A., Hinton, G.E. (2008). A scalable hierarchical distributed language model. Advances in Neural Information Processing Systems, 21.

[12] Bengio, Y., Ducharme, R., Vincent, P. (2000). A neural probabilistic language model. Advances in Neural Information Processing Systems, 13. https://jmlr.csail.mit.edu/papers/volume3/bengio03a/bengio03a.pdf, accessed on Jan, 2, 2023.

[13] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P. (2011). Natural language processing (almost) from scratch. Journal of machine learning research, 12(ARTICLE): 2493-2537.

[14] Joulin, A., Grave, E., Bojanowski, P., Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.
[15] Wang, W., Yang, N., Wei, F., Chang, B., Zhou, M. (2017). Gated self-matching networks for reading comprehension and question answering. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, 1: 189-198. http://dx.doi.org/10.18653/v1/P17-1018
[16] Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
[17] Turkson, R.F., Yan, F., Ali, M.K.A., Hu, J. (2016). Artificial neural network applications in the calibration of spark-ignition engines: An overview. Engineering Science and Technology, an International Journal, 19(3): 1346-1359. https://doi.org/10.1016/j.jestch.2016.03.003
[18] Eichie, J.O., Oyedum, O.D., Ajewole, M.O., Aibinu, A.M. (2017). Artificial Neural Network model for the determination of GSM Rxlevel from atmospheric parameters. Engineering Science and Technology, an International Journal, 20(2):795-804. https://doi.org/10.1016/j.jestch.2016.11.002
[19] Ozer, I., Ozer, Z., Findik, O. (2018). Noise robust sound event classification with convolutional neural network. Neurocomputing, 272, 505-512. https://doi.org/10.1016/j.neucom.2017.07.021
[20] Aghzal, M., Mourhir, A. (2021). Distributional word representations for code-mixed text in Moroccan Darija. Procedia Computer Science, 189: 266-273. https://doi.org/10.1016/j.procs.2021.05.090
[21] Liao, Z., Ni, J. (2021). Construction of Chinese synonymous nouns discrimination and query system based on the semantic relation of embedded system and LSTM. Microprocessors and Microsystems, 82, 103848. https://doi.org/10.1016/j.micpro.2021.103848
[22] Mahdaouy, A.E., Gaussier, E., Alaoui, S.O.E. (2016, October). Arabic text classification based on word and document embeddings. In International Conference on Advanced Intelligent Systems and Informatics, pp. 32-41.
[23] IRCAM. (n.d.). Institut Royal de la Culture Amazighe. https://www.ircam.ma/.