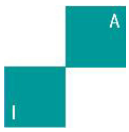
Inteligencia Artificial, 24(67), 1-17
doi: 10.4114/intartif.vol24iss67pp1-17



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

Morphological Skip-Gram: Replacing FastText characters n-gram with morphological knowledge

Flávio Arthur O. Santos
faos@cin.ufpe.br
Universidade Federal de Pernambuco, Centro de Informática, Recife, Brazil.

Hendrik Teixeira Macedo
hendrik@dcomp.ufs.br[orcid=0000-0002-6477-756X]
Universidade Federal de Sergipe, Departamento de Computação, São Cristóvão, Brazil

Thiago Dias Bispo
tdb@cin.ufpe.br
Universidade Federal de Pernambuco, Centro de Informática, Recife, Brazil.

Cleber Zanchettin
cz@cin.ufpe.br[orcid=0000-0001-6421-9747]
Universidade Federal de Sergipe, Departamento de Computação, São Cristóvão, Brazil

Abstract Natural language processing systems have attracted much interest of the industry. This branch of study is composed of some applications such as machine translation, sentiment analysis, named entity recognition, question and answer, and others. Word embeddings (i.e., continuous word representations) are an essential module for those applications generally used as word representation to machine learning models. Some popular methods to train word embeddings are GloVe and Word2Vec. They achieve good word representations, despite limitations: both ignore morphological information of the words and consider only one representation vector for each word. This approach implies the word embeddings does not consider different word contexts properly and are unaware of its inner structure. To mitigate this problem, the other word embeddings method FastText represents each word as a bag of characters n-grams. Hence, a continuous vector describes each n-gram, and the final word representation is the sum of its characters n-grams vectors. Nevertheless, the use of all n-grams character of a word is a poor approach since some n-grams have no semantic relation with their words and increase the amount of potentially useless information. This approach also increase the training phase time. In this work, we propose a new method for training word embeddings, and its goal is to replace the FastText bag of character n-grams for a bag of word morphemes through the morphological analysis of the word. Thus, words with similar context and morphemes are represented by vectors close to each other. To evaluate our new approach, we performed intrinsic evaluations considering 15 different tasks, and the results show a competitive performance compared to FastText. Moreover, the proposed model is 40% faster than FastText in the training phase. We also outperform the baseline approaches in extrinsic evaluations through Hate speech detection and NER tasks using different scenarios.

Keywords: Natural Language Processing, Word Embeddings, Morphological Knowledge, Character n-grams

1 Introduction

Natural language processing (NLP) is a branch of Machine Learning that helps computers understand, interpret, and manipulate human language allowing applications to read, hear, interpret it, measure

ISSN: 1137-3601 (print), 1988-3064 (on-line)

sentiment and deal with unstructured data.

Most of the knowledge generated today is unstructured, from medical records to social media and extract and understand these pieces of information demand intelligent systems able to perform complex behavior.

Deep Learning (DL) models have achieved the state-of-the-art results in many Natural Language Processing (NLP) tasks as machine translation [41], question and answering (Q&A) [23], image captioning [23], text summarization [30], named entity recognition [21] and sentiment analysis [22]. Most of these solutions use Word Embeddings (WE) to represent the input as continuous word vectors. Each word has one-word embedding. WE are fundamental to achieve good results in NLP tasks.

Word embedding's primary goal is to serve as word representation to be given as input for the machine learning model. Thus, they need to represent the maximum word information as possible. There is semantic and syntactic information, such as synonyms, antonyms, radical, lemma, stemming, morphological knowledge, part-of-speech, and others. Although the word embeddings need to represent most word information as possible, its training can not be computationally costly because it will be used in an expert system. Thus, there is a trade-off between being informative and not be computationally expensive.

There are three main approaches to built WE: (i) Context-window based models, (ii) Semantic relationship-based models, and (iii) Graph Distance-based models. Each method has drawbacks, though; models from (ii) and (iii) use knowledge bases such as WordNet [28] and Freebase [7] to learn the WE, but consider only a tiny part of the dataset. Models from (iii) use mainly the Leacock-Chodorow [9] distance to capture the semantic relationship between two words, ignoring alternative graph distances. Finally, knowledge bases involved are often limited to specific fields. Some models from (i), such as Neural Language Model [11] and Word2Vec [26], despite the good results, are trained using only local context information of every word instead of global context information. The Word2Vec method also does not use the internal word structure information (Morphology information). Although GloVe [32] uses global word context information, words with different lexical but equal meaning (paraphrases) have different representations because they are in a diverse global context. The FastText [6] model, based on Word2Vec, proposes the use of the internal word structure based on a bag of all n-gram characters of each word. Besides being a brute-force solution, some character n-grams have no semantic relationship with the formed word; the word 'American' and their character n-gram 'erica', for instance, have no semantic connection at all.

We propose the Morphological Skip-Gram (MSG) model, replacing the bag of characters n-grams with a bag of morphemes. A morpheme is the smallest grammatical unit in a language. Thereby, words with common morphemes will have a similar representation. Our approach is important because it allows the uses of the word inner structure that has a syntactic relation with the complete word. This sounds like a more consistent scientific hypothesis than that of FastText considering grammatically well-behaved texts.

The rest of this paper is organized as follows. In section 2, we depict the word embeddings technique. Section 3 presents our approach to word representation using the bag of morphemes. Experiments for intrinsic and extrinsic evaluation are presented and discussed in section 4. We conclude the work in section 5.

An example of morphological analysis performed by Morfessor toolkit is: given the Portuguese word *federativas*, it returns the set of morphemes = $\{federa, tiva, s\}$ as output.

2 Word Embeddings

This section discusses some methods to train word embeddings.

[26] proposed two architectures to learn word embeddings considering the word context window of every word into the corpus. The methods are Skip-Gram and CBOW. The CBOW aims to predict the central word c based on its context, while Skip-Gram gave a sentence s and a central word c, predict the possible words in the context of c.

Formally, the Skip-Gram goal is to maximize the function E_{sk}:

$$E_{sk} = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\geq c, j\neq 0} \mathbf{log} p(w_{t+j}|w_t) \quad (1)$$

Where c is the context window size. Once to compute $p(w_{t+j}|w_t)$ is computationally expensive, the authors proposes the use Negative Sample [26] and approximate $\mathbf{log}p(w_{t+j}|w_t)$ according with equation 2.

$$\mathbf{log}\sigma(w_{t+j}^T w_t) + \sum_{i=1}^{k} E_{w_t \sim P_n(w)}[\mathbf{log}\sigma(-w_i^T w_t)] \tag{2}$$

Thus, the objective is to distinguish the target word w_{t+j} of the words taken from a noise distribution $P_n(w)$, where k is the number of negative samples for each target.

[31] proposes the GloVe model. In contrast with Skip-Gram and CBOW, the GloVe model uses the corpus global statistics to learn the word embeddings. Thus, Let X be a co-occurrence matrix among the words, where X_{ij} indicates how many times the word j appear in word i context. The GloVe objective is to minimize the following cost function:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - log(X_{ij}))^2 \tag{3}$$

Where V is the vocabulary size, w_i the word embedding of the central word i, and $\tilde{w}_j$ is the word embedding of the context word j. As we can see in the equation 3, the GloVe aim is to approximate the inner product between the vectors w_i and w_j to log the co-occurrence between them. [38] presents a method complementary to the GloVe, whose objective is to uses the Paraphrase Dataset [16] to complete the X cooccurrence matrix.

[6] proposed a new approach based on Skip-Gram. In this approach, each word is a set of characters n-grams. Every character n-gram is associated with a continuous vector representation. Thus, each word is a sum of all character n-gram representations.

[24] incorporates explicitly morphological knowledge in word embeddings. Each word i has a vector v_i and its morpheme vectors. After obtained the new representations of each word, the method uses the Natural Language Model [11] to lean the word embeddings.

[42] proposed a method to incorporate morphological knowledge into word embeddings implicitly. Each word i is a vector v_i and the set M_i, which corresponds to the morpheme meanings. To build M_i, first, the authors extract the suffix, prefix, and root of the word i and adds to the set M_i. After obtaining all new words representations of a vocabulary V, they use the CBOW method to learn the word embeddings.

[2] present a study about how different types of linguistic information (surface form, lemma, morphological tag) affect the semantic and syntactic relation of word embeddings. The authors consider three sets of information: W, L, and M. W is a set of the word surface forms, L is the set of lemma and M a set of morphological tags. Thus, each word i in the vocabulary V is a vector v_i added the W, L, and M representations. After obtaining this new representation, the authors use Skip-Gram to learn the word embeddings.

[34] explicitly incorporates the morphological knowledge in the word embeddings. In the first step, they perform morphological analysis of each word i in vocabulary V. Thus, each word i is a sum of its vector v_w and its morpheme vectors. The authors use CBOW to learn the word and morpheme representations. It is essential to highlight that the authors gave a weight of 0.8 to vector v_i and 0.2 to the morpheme vectors.

[12] proposed an extension of the Log-Bilinear model to learn word embeddings. The proposed variation, called Morpho-LBL, consists of adding a multi-objective to the LBL model to optimize the model so that it predicts the next word and its respective morpheme.

[13] introduces morphological knowledge into the neural machine translation connection. The authors investigate which part of the decoder is best to add morphological knowledge.

[34] proposed an approach to incorporate morphological knowledge into CBOW architecture. Each word is represented by a token identifying it uniquely and the tokens of its morphemes. Similar to [34], [37] incorporates the words morphological category during word embeddings training. They use CBOW as the base model.

ELMo [33] is a model based on n-gram characters. Their representations are computed through two independent LSTM [18], in which one analyzes the context on the right (step forward) and the other

analyzes the context on the left (step backward). The final representation of the word consists of the weighted sum between all the intermediate representations.

BERT [14] uses an attention mechanism [40] to learn the word contextual representation, a characteristic that allows determining which of the contextual information is most relevant for final word vector representation. Unlike traditional word embedding and ELMo techniques, BERT considers the order of neighboring words on the left and right simultaneously. Like ELMo, words can have different vector representations depending on the context in which they are inserted.

This section presented different state-of-the-art methods to train word embeddings. Considering the methods approach, GloVe [32], CBOW, and Skip-Gram [26] views every word as a unique token and do not uses the inner structure information (morphological knowledge) during the word embedding training. [42] presents a method to add morphological knowledge into the CBOW architecture, but they give a weight of only 0.2 to morphological information. FastText [6] is a method based on Skip-Gram and considers the inner structure word information. Still, it is a computationally expensive approach because it uses all character n-grams of every word in the vocabulary.

3 Morphological Skip-Gram (MSG)

The proposed approach aims to incorporate morphological knowledge in the Skip-Gram method. We first present the morphological analysis and the baseline model FastText. Next, we detail our proposed method MSG. Lastly, we point out the differences between FastText and MSG.

3.1 Morphological Analysis

Morphological analysis is the task of finding the morphemes of a word. The morpheme is the smallest unit that carries the meaning of a word. The morphemes obtained by the morphological analysis are classified in (i) **Radical/base**: part common to a certain set of words, from which other words will be formed; (ii) **Gender and number ending**: has the function of indicating whether the word is in masculine or feminine, plural or singular; (iii) **Thematic vowel**: links the radical to the endings (terminal elements indicative of the inflections of words) that form the words, constituting the theme; (iv) **Affixes (prefixes and suffixes)**: prefixes are the particles that are located before the radical and suffixes appear afterward;

3.2 FastText

The Skip-Gram method uses a different vector to each word and ignores information about the inner word structure. The FastText model proposes a score function considering the inner word structure information. In FastText, each word w is represented as a bag of n-grams characters. The full word w is also added in the bag allowing the model to learn the continuous vectorial representations of the words and their characters n-grams. Using the Portuguese word $federativas$ as an example, the 4-gram and 5-gram characters generate the following bag of tokens:

<fede, eder, dera, erat, rati, ativ, tiva, ivas, feder, edera, derat, erati, rativ, ativa, tivas>, <federativas>

In practice, the FastText model uses the characters n-grams of size 3, 4, 5, and 6.

Supposing that we have a dictionary G of characters n-grams, where $|G| = K$. Given the word w, we denote $G_w \subset G$ as the set of characters n-gram present in w, and each n-gram g in G is associated with a continuous vectorial representation named as z_g. Thus, the word w is represented by the sum of all vectorial representation of its characters n-gram (z_g). Finally, we obtain the following score function used to predict when the word c appear in the context of w:

$$s(w, c) = \sum_{z_g \in G_w} z_g^T v_c \tag{4}$$

Rewriting the objective function of Skip-Gram using the FastText score function, we obtain:

$$E_{fasttext} = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \mathbf{log}(p(w_{t+j}|w_t)) \tag{5}$$

Where $\mathbf{log}p(w_{t+j}|w_t)$ is calculated through:

$$\begin{aligned}\mathbf{log}(p(w_{t+j}|w_t)) &= log(\sigma(s(w_{t+j}, w_t))) \\ &+ \sum_{i=1}^{k} E_{k_t\sim P_n(w)}[log(\sigma(-s(w_i, w_t)))]\end{aligned} \tag{6}$$

As shown above, word representation considers the character's n-gram representation. As a consequence, the new FastText score function also incorporates the context word information in the vectorial representation of the characters n-gram and consequently in the considered words.

3.3 Morphological Skip-Gram

Our proposal uses the morphology of the words as part of the full word representation. The aim of the Morphological Segmentation Task is to segment words in its morphemes, the smallest meaning-carrying unit of a word. For each word v in the vocabulary V, we define a set m_v, where:

$$m_v = \{x \mid x \text{ is a morpheme of v}\} \tag{7}$$

We used Morfessor toolkit [39] to build the set m_v. The Morfessor is a popular toolkit for statistical morphological segmentation. It is composed of a family of unsupervised learning methods.

In the original Skip-Gram, each word v is represented by only one vector w_v. In our proposal, each word is represented as:

$$k_v = w_v + \sum_{x}^{m_v} z_x \tag{8}$$

Where z_x is the vectorial representation of each morpheme x of word v. Thus, as in Skip-Gram, the MSG also has two continuous representations for each word. The first one considers the word as the center of the sentence. The second representation of the word is the context.

The MSG aim is to maximize the function E_{msg}:

$$E_{msg} = \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\geq c, j\neq 0} \mathbf{log}p(k_{t+j}|k_t) \tag{9}$$

We also use Negative Samples through training optimizing $\mathbf{log}p(k_{t+j}|k_t)$ according to equation 10.

$$log\sigma(k_{t+j}^T k_t) + \sum_{i=1}^{k} E_{k_t\sim P_n(w)}[log\sigma(-k_i^T k_t)] \tag{10}$$

As we discussed before, some methods such as GloVe [32], Skip-Gram [26], CBOW [26] present interesting results, but they do not use the inner structure word information. Our proposed method, MSG, overcomes this limitation adding morphological word knowledge into the Skip-Gram architecture. Besides, every word morpheme token (z_x in equation 8) has equal weight to the word token (m_v in equation 8), differently from approach present in [42] that add morphological knowledge into CBOW architecture and give different weights to morpheme vectors and word vectors.

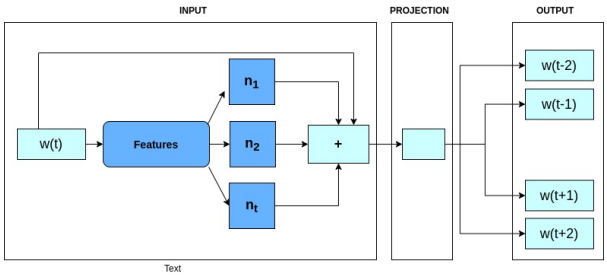


Figure 1: FastText and MSG general model

3.4 Differences between the FastText and the Morphological Skip-Gram

The FastText method is our baseline; thus, this section will focus on discussing the difference between our proposed method and it. Figure 1 present a visual representation of the Morphological Skip-Gram and FastText architectures in a general way. In order to simplify the figure, we use only a context of size 2 (the output, thus, has only four words): $w(t-2)$, $w(t-1)$, $w(t+1)$ and $w(t+2)$, two previous and two posterior words related to the central word $w(t)$.

The main difference between the two models is the source of information added in the Skip-Gram architecture, represented in figure 1 as a feature block. The feature block output in MSG is all word morphemes, while in FastText are all characters n-grams. An important point to highlight is the complexity of the two models, which is dependent on the number of tokens produced in the feature block. In the training step of both, it is necessary to compute the derivatives concerning each one of vectorial representations (morphemes in the Morphological Skip-Gram and characters n-grams in the FastText). Thus, in the example of the Portuguese word $federativas$, using the sets of 4-gram and 5-gram of characters, we need to learn 16 representations in the FastText model. However, using the set $\{federa, tiva, s\}$ obtained from the morphological analysis, we need to learn four representations in the MSG model. That example shows up the huge difference between uses morphological knowledge as inner structure information and all character n-grams as inner structure information.

Yet from the perspective of the architecture, we can make an analogy with an artificial neural network composed of three layers: input, projection, and output. The projection layer represents the word embeddings, morphemes, or n-grams; the more substantial dimension in this layer results in more learned parameters and hence a larger space to represent information.

4 Evaluation and discussion

We performed (i) intrinsic and (ii) extrinsic evaluation. For intrinsic evaluation, we used universal assessment methods well known by the scientific community, whereas in (ii), we used practical case studies.

4.1 Intrinsic evaluation

Intrinsic evaluation measure the word embeddings quality compared to human judgment. We used 15 well know datasets to perform the intrinsic evaluation. The datasets fall into three categories: (i) Similarity, (ii) Analogy, and (iii) Categorization.

- **Similarity**: datasets composed of pairs of words, where each pair has an average rank defined by humans. This category consists of 6 datasets: SimLex999 [17], MEN [8], WordSimilarity353 [15], Rare Words [15], RG65 [36] e Turk [35].
- **Analogy**: datasets composed of two pairs of words relative to a specific relation. For example, (man, woman), (king, queen). This category is composed of 3 datasets: Google [25], MSR [27], SemEval 2012.2 [20].

- **Categorization**: problems involving a sentence or word with a binary target. This category is composed by 3 datasets: AP [1], BATTING [5], BLESS [4].

We use the toolkit developed by [19] to evaluate the word embeddings in these 15 datasets.

4.1.1 Corpora and training details

The corpus 1 billion words language model benchmark [10] was used to train all the word embeddings. This corpus consists of approximately 1 billion words. We do not perform the pre-processing step because the corpus is ready to use.

Table 1: Training details

PARAMETERS	**VALUES**
dimensions	50, 100, 200, 300
iterations	20
threads	12
negative sample	5
context window	5
learning rate	0.1

Table 1 shows the parameter values used along with the training of all word embeddings. It is important to note that the values of threads, negative samples, context window, and learning rate were chosen according to the default values of the FastText project.

4.1.2 Results and discussion

In these experiments, we only compare MSG with FastText because, in FastText paper, the authors already made a comparison between FastText and other word embeddings models and achieved better or competitive results. There are recent deep learning-based models that learn good word representations, such as BERT and ElMo, but BERT and ELMo are context-sensitive models; the same word has different representations depending on its context. However, FastText, GloVe, Word2Vec, Morphological Skim-Gram are context insensitive because, after training, the word has the same representation no matter its context. Thus, these word embeddings models are in different categories, and we do not consider a fair comparison.

Table 2: Summary of datasets and metrics

Dataset	Category	Metrics
SimLe999	Similarity	$\rho - Spearman$
MEN	Similarity	$\rho - Spearman$
Word Similarity 353	Similarity	$\rho - Spearman$
Rare Words	Similarity	$\rho - Spearman$
RG 5	Similarity	$\rho - Spearman$
Turk	Similarity	$\rho - Spearman$
Google	Analogy	Accuracy
MSR	Analogy	Accuracy
SemEval 2012.2	Analogy	Accuracy
AP	Categorization	Purity
BLESS	Categorization	Purity
BATTING	Categorization	Purity

In experiments, we used a Dell desktop with the 8th generation Intel core i5 processor, 8GB of RAM, 1TB of storage, running the Ubuntu 16.04 operating system. We ensure that both models are executed

in the same environment situation to obtain the values of Table 6. Table 2 presents a summary of all used datasets in the intrinsic evaluation, highlighting the dataset, category, and used metric. High values represent better results.

Tables 3, 4, and 5 present the results of the experiments performed using tasks of analogy, similarity and categorization, respectively. Table 6 shows a comparison of training time between the models. The name ft-d50 means that these word embeddings were trained using FastText (FT) and have dimension 50. While msg-d50 means it was trained using the Morphological Skip-Gram (MSG) and has dimension 50.

Table 3: Analogy results

Name	**Google**	**MSR**	**SE-2012**
ft-d50	0.000	0.001	0.12747
msg-d50	0.005	0.001	0.128
ft-d100	0.008	0.004	0.143
msg-d100	0.064	0.0126	0.144
ft-d200	0.087	0.051	0.155
msg-d200	0.242	0.104	0.164
ft-d300	0.128	0.082	0.168
msg-d300	0.332	0.211	0.180

The MSG model presented superior results to the FT in all datasets and all embeddings dimensions. The except case was the dataset MSR using embeddings ft-d50 and msg-d50, in which both had the same performance (Table 3). However, embeddings of dimension 50 did not have a good performance on all three datasets. We observed a variation of the results with the change of the size of the embedding. For instance, in the results using the Google dataset, the msg-d50 model obtained 0.005 while the msg-d300 obtained 0.332, in other words, the size of the embeddings has much influence on the model performance at Google Analogy benchmark.

Table 4: Similarity Results

Name	**SL9**	**MEN**	**WS353**	**RW**	**RG65**	**Turk**
ft-d50	0.25	0.64	0.61	0.25	0.58	0.63
msg-d50	0.25	0.64	0.61	0.23	0.56	0.63
ft-d100	0.29	0.69	0.64	0.28	0.65	0.66
msg-d100	0.30	0.68	0.64	0.25	0.64	0.63
ft-d200	0.33	0.71	0.65	0.30	0.71	0.67
msg-d200	0.33	0.71	0.65	0.29	0.67	0.64
ft-d300	0.34	0.73	0.66	0.31	0.73	0.67
msg-d300	0.35	0.71	0.66	0.29	0.68	0.66

From table 4, we can see the MSG model presented competitive results compared to FT. The methods showed a difference of a maximum 0.05 (RG65 dataset using the ft-d300 and msg-d300 models). It is essential to point out that the FT was the best in all the Rare Words (RW) dataset cases, generalizing better in the unknown words scenario. It is important to highlight that FastText is designed to deal with word out of vocabulary, rare words, and word with spelling errors. This FastText characteristic is due to characters n-grams because even if a word is out of vocabulary, some (or all) of its characters n-grams can be in n-grams vocabulary. Thus, since the RW scenario is composed of words with low frequency, some of its characters n-grams can be present in word with high frequency.

Considering the Table 5 results, both models presented similar performance, differentiating by a maximum of 0.01.

Considering Tables 3-5 results, the MSG model is better than the FT in the analogy tasks, whereas, in the Similarity and Categorization evaluations, the models had comparable performance. A possible explanation of why the MSG model presents better results than FT in analogy task is that some tokens of the FT brute force (all characters n-gram) solution may be adding noise in the word representation. Thus,

Table 5: Categorization Results

Name	AP	BLESS	BATTING
ft-d50	0.61	0.75	0.43
msg-d50	0.62	0.74	0.42
ft-d100	0.60	0.77	0.43
msg-d100	0.59	0.77	0.45
ft-d200	0.61	0.80	0.43
msg-d200	0.60	0.81	0.43
ft-d300	0.59	0.81	0.43
msg-d300	0.60	0.82	0.42

Table 6: Training Time

Name	AP
ft-d50	307m27
msg-d50	196m11
ft-d100	409m42
msg-d100	268m50
ft-d200	647m32
msg-d200	414m30
ft-d300	853m58
msg-d300	541m35

since the morphological analysis only adds expert linguistic knowledge, it produces only the necessary tokens (morphemes) to represent the inner word structure.

Table 6 shows that the training time of the MSG model is approximately 40% faster than the FT model. That improvement in training time is expected because the MSG model uses expert knowledge information to represent the inner structure of the word, unlike FastText, which uses a force brute solution.

From the results obtained in intrinsic evaluation, we can see that Morphological Skip-Gram achieved competitive results compared with FastText and is approximately 40% faster than it. Both methods have the Skip-Gram as base architecture, being different on the inner structure information: MSG uses morphemes of the word, and FastText uses all characters n-grams. Thus, there is strong evidence that the morphological information is sufficient to represent the inner structure word information, and the FastText brute force solution has some characters n-gram, which are not useful.

4.2 Extrinsic Evaluation

Extrinsic evaluation measures the word embeddings quality when it is used as feature vectors in a supervised machine learning task. We used two NLP tasks to evaluate the proposed model: (i) Named Entity Recognition [21] (in Portuguese), and (ii) Hate Speech Detection [3] (in English). We chose these two tasks because the first one evaluates the model's ability to classify a word against a context. The second task demands the model to represent one single sentence (or a paragraph) and classify it.

4.2.1 Named Entity Recognition

Named Entity Recognition (NER) is an NLP task to identify and classify entities in a given text. Some categories of entity are: "Pessoa" (person), "Local" (location), "Organização" (organization), "Valor" (value), and "Tempo" (time). This task is critical in the process of text information extraction. To solve this problem, the model must receive as input the word representation in a given sentence and return the entity or the information that it is not an entity. In the following tables, "LOC" and "ORG" are abbreviations to Localization and Organization classes, respectively.

Figure 2 shows a visual representation of the NER architecture model used in our experiments. This architecture was presented in the work of [21]. The vector $w = [w_1, w_2, ..., w_n]$ represents the word

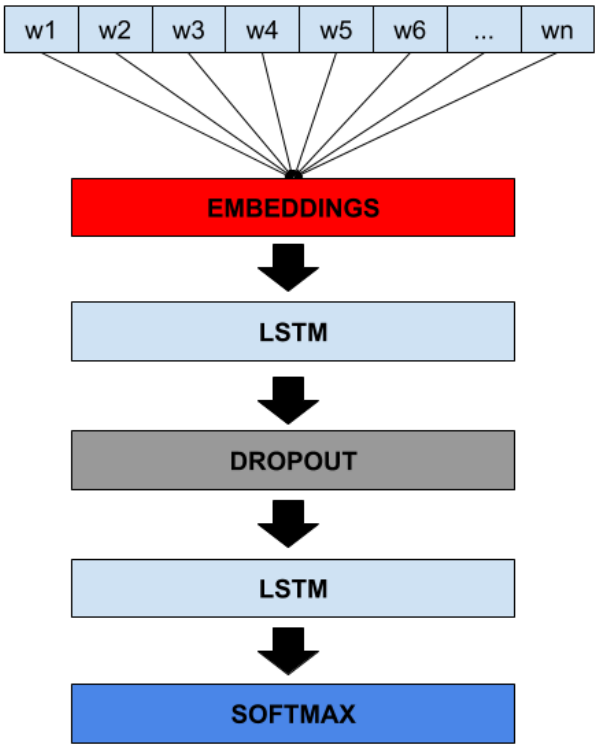


Figure 2: Graphical representation of the NER model architecture.

information, where w_i is the index of a word. This vector contains three information about the word to be classified: (i) left context, (ii) word index, and (iii) right context. For instance, using the context window of size 3, the vector w has seven positions. The layer *embeddings* receives the vector w as input and returns a matrix M as output, on which each line i of M represents the *word embedding* of the word w_i. The weights of this layer may be initialized using the *word embedding* learned by any models presented in Word Embeddings section, allowing the neural network to use a pre-trained knowledge. Then, because it is a sequence of a classification problem, the architecture uses two LSTM layers with a Dropout layer between them. The model output uses a dense layer using the *softmax* function to returns the classification probabilities.

Experiments and results To initialize the models *embeddings* layer weights, we used the *word embeddings* trained with the Morphological Skip-Gram model. This experiment was split into two scenarios. In the first one, we used the dataset PrimeiroHarem in the training phase and the MiniHarem to test. In the second scenario, we used the dataset Paramopama (here, also referenced as "Param") to training and the final 10% of WikiNER to test. These two scenarios were defined according to the work of [21].

Table 7: *Datasets* size

Dataset	**Sentences**	**Tokens**	**Type**	**Sceario**
PrimeiroHarem	4,749	93,125	Train	1
MiniHarem	3,393	62,914	Test	1
Paramopama	12,500	310,000	Train	2
WikiNER (10%)	5,855	149,613	Test	2

Table 7 presents the used datasets information. The table 8 shows the number of tokens per class of each dataset, turning explicit that all used datasets are unbalanced. Moreover, it is important to point out the classifier of the first experiment is trained with six classes ('Outro', 'Pessoa', 'Local', 'Organization',

'Tempo', and 'Valor'), while the second scenario is trained with five classes. The class 'Outro' means that the word is not a named entity.

Table 8: *Datasets* distribution

Entities	PrimeiroHarem	MiniHarem	Paramopama	WikiNER
Other	86,682	59,023	263,916	122,671
Person	2,242	1,651	7,326	5,510
LOC	2,036	1,385	17,461	6,536
ORG	2,168	1,372	7,154	10,827
Time	1,420	701	10,827	6,506
Value	1,121	712	0	0

The architecture of the Paramopama-WNN (Param-WNN) model is present in Figure 2. The Paramopama-CWNN (Param-CWNN) model is similar to the Figure 2, however, it also uses a convolutional layer to extract information from the characters (*char embeddings*), using, thus, words and its characters information. The performance of Param-CWNN and Param-WNN were taken from [21].

Table 9: Results (%)- Scenario 1.

Model	Precision	Recall	F-measure
Param-MSG	74.10	**74.59**	**74.07**
Param-CWNN	**75.13**	68.38	71.35
Param-WNN	73.68	69.26	71.22

In order to compare our results with the *baseline* model, we used the same scenarios and metrics presented in [21]. Tables 9-10 show the results of the Scenario 1. We can note that the use of the morphological *word embeddings* had an important influence since the model Paramopama-MSG (Param-MSG) obtained 74.07% of F-measure against 71.22% of the model Param-WNN. Another important aspect is the F-measure result of "Tempo" class: Param-MSG obtained 70.00% while the Param-WNN result is 60.77%. Our assumption to explain this performance is that since these entity words are not similar (it represent dates, month or time of the year), the context word morphemes of the entity are similar, so the morphological information implicitly present in *word embeddings* might have helped.

The results of Scenario 2 are presented in the Tables 11-12. As in Scenario 1, the model using morphological *word embeddings* obtained better results; however, in this scenario, it was only a slight improvement. Again, the result obtained by Param-MSG in the "Tempo" class had the best performance, contributing to our assumption that the morphemes present in the context of the words of the "Tempo" class are similar. A future investigation is to evaluate the types of writings of the Paramopama datasets, WikiNER, PrimerHarem, and MiniHarem to verify the contribution of the use of the morpheme information.

4.2.2 Hate speech

Hate speech refers to words, phrases, or expressions that insult, intimidate or harass people considering race, color, ethnicity, nationality, sex or religion, or who can instigate violence, hate, or discrimination against people [29]. The automatic detection of such content aims to determine whether or not there is

Table 10: Results by entities (%)- Scenario 1.

Model	Param-MSG	Param-CWNN	Param-WNN
Person	**77.26**	75.95	76.87
LOC	**70.13**	68.57	68.75
ORG	**55.28**	54.22	52.07
Time	**70.00**	55.00	60.77
Value	73.69	**75.81**	70.85

Table 11: Results (%)- Scenario 2.

Model	Precision	Recall	F-measure
Param-MSG	**88.45**	88.68	**88.49**
Param-CWNN	83.97	78.39	80.50
Param-WNN	86.45	**89.77**	88.08

offensive content within texts or even determine its type. Some of its most common types are misogyny, racism, xenophobia, homophobia, discrimination by appearance, discrimination by political ideology, and so on. In the following, we will explain the model architecture used for this NLP task.

Model In this work, we used the best model architecture proposed by [3]. The training flow, from the input layer to its classification, follows the order of the steps described by the reference work, namely: i) **Preprocessing and vectoring** through the use of dataset vocabulary word indexes; ii) **Input layer** or the embeddings generation/training layer; iii) **LSTM** layer; iv) **Dense** layer.

4.2.3 Experiments and Results

The dataset used in the experiments is the same as [3] and consists of 16,131 tweets labeled as 'sexism', 'racism' or 'none'. Not all tweets were available at the time of download and, therefore, our dataset is less than that considered in the reference work.

In our experiments, we used an LSTM combined with a Gradient Boosting Decision Tree (GBDT), which represents the best architecture validated by [3]. When we combine the LSTM model with the GBDT, we ignore the result of the neural network using the dense layer and performing a new training with the GBDT using the embeddings learned in the LSTM layer as features. This process is according to the reference work.

The configuration of the model parameters was as follows: maximum vector size expected by the input layer: 10,000; the size of generated embeddings: 200; First and second layer dropout rate: 0.25 and 0.50, respectively. LSTM optimization function: RMSProp; Error Function: Categorical Cross-Entropy; Training bach size: 128.

Like the reference work, for each of these architectures, we performed two experiments: In the first one, we set the weights of the input layer using the GloVe pretrained word embeddings used in the referenced article. In the second, the weights were the embeddings of the Morphological Skip-Gram (MSG) model extracted from the same dataset. Both models were cross-validated with ten folds each, and the metrics calculated by averaging their values in each fold. Therefore, the f-measure value may be outside the range defined by the accuracy and recall. The results are listed in Table 13. Baseline 1 (line 1) and Baseline 2 (line 4) are the values obtained by [3] in the execution of the same experiments with LSTM and LSTM + GBDT, respectively. Its initialization used GloVe, as explained before.

The lines 2,3 corresponds to our experiments considering the Baseline 1, and lines 5,6 corresponds to our experiments considering the Baseline 2.

As we can see, by initializing the LSTM architecture model with the trained *embeddings* using the MSG model, we get values slightly lower than those obtained when the initialization using the embeddings trained with GloVe. With the MSG, we obtained the results of 0.829, 0.823, and 0.818, and with GloVe, we got 0.833, 0.831, and 0.827 of accuracy, recall, and f-measure, respectively. In contrast, using the LSTM model combined with the GBDT as described before, the scenario reverses, and the results using MSG are superior to those obtained with GloVe: 0.910, 0.911, 0.910 versus 0.905, 0.906, 0.905 accuracy, recall, and f-measure, respectively.

Table 12: Results by entities (%)- Scenario 2.

Model	Person	LOC	ORG	Time
Param-MSG	**89.06**	87.78	**75.65**	**91.33**
Param-CWNN	87.45	83.57	62.73	71.00
Param-WNN	87.00	**87.82**	75.41	88.00

Table 13: Result of experiments with hate speech dataset.

#	Architecture	Init.	Precision	Recall	F-measure
1	**Baseline 1**	GloVe	0.807	0.809	0.808
2	LSTM	GloVe	**0.833**	**0.831**	**0.827**
3	LSTM	MSG	0.829	0.823	0.818
4	**Baseline 2**	GloVe	0.849	0.848	0.848
5	LSTM + GBDT	GloVe	0.905	0.906	0.905
6	LSTM + GBDT	MSG	**0.910**	**0.911**	**0.910**
7	**Baseline 3**	FastText	–	–	–

The results presented in the Table 13 demonstrate that the use of the morphemes embeddings in the addressed models reached similar values when compared to the results with the well-established GloVe, exceeding its performance in some cases.

In the same way as in the experiments with Named Entities Recognition, our experiments reached superior results to those of the reference work, as we can observe in experiments 1 and 4 of Table 13. Although the datasets were of different sizes, there is strong evidence that the use of the embeddings produced from the Morphological Skip-Gram model was responsible for the performance improvement. This was the only difference between our experiments and [3].

4.3 Morphological Skip-Gram Limitations

The intrinsic evaluation shows that our proposed method, MSG, presents competitive results and is about 40% faster than our baseline FastText. Furthermore, the extrinsic evaluation shows that when we use our MSG embeddings, the deep neural networks present better results than our baselines. However, even presenting excellent results in intrinsic and extrinsic evaluations, the MSG has limitations. For instance, we only consider the Skip-Gram architecture to add morphological knowledge; there are other architectures, such as GloVe and Neural Language Model, that we can introduce morphological knowledge. Besides, there is other syntactic expert knowledge to be added in word embeddings, such as stemming, radical, and lemma.

5 Conclusion

This work presented the Morphological Skip-Gram (MSG) method to learn word embeddings using expert knowledge, especially the morphological knowledge of words. The MSG uses the morphological structure of words to replace the n-grams bag of characters used in the FastText model. The use of such a bag of n-grams is a brute force solution as it tries all possible combinations of characters n-grams of a word. As the purpose of using this bag of n-grams is also to learn the information about the internal structure of the words, we consider the morphological information more robust and informative. We compared MSG with FastText in 12 benchmarks. Results show that MSG is competitive compared to FastText and takes 40% less processor time than FastText to train the word embeddings. Keeping the quality of word embeddings and decreasing training time is very important because usually, a corpus to training embeddings is composed of 1B tokens. The Common Crawl corpora contain 820B tokens, for example, [31].

From a technological perspective, our extrinsic evaluation results show that when we use the MSG embeddings, the deep neural networks trained to achieve better results than our baseline results. Thus, showing strong evidence that the morphological information of the MSG embeddings is essential.

5.1 Future Works

The proposed approach opened some future research considering:

Morpheme representations as explicit features: The applications in extrinsic evaluation used just the final word representation obtained by MSG (equation 8). However, future works are necessary

to develop new approaches to use the morphemes representation explicitly. A possible approach is to use attention models to give an importance score for each morpheme representation.

Morphological analyzer: Since the morphological analysis tool used is not exact and has a fundamental impact on the MSG model. It is necessary to experiment using other morphological analysis tools to observe the impact on the quality of the generated word embeddings.

Expert knowledge : In addition to the morphological analysis, it is possible to explicitly add other specialist knowledge during the training of word embeddings. For example, we can add the lemma, stemming, and stem information of the word so that words with this information in common have close representations.

GloVe morphological: GloVe is a word embeddings method widely used in the literature. However, it does not explicitly learn information about the internal structure of words. Hence, an interesting research direction of this work is to study a way to introduce the morphological knowledge of words in GloVe.

Acknowledgements

The authors thank CAPES and FAPITEC-SE for the financial support [Edital CAPES/FAPITEC/SE No 11/2016 - PROEF, Processo 88887.160994/2017-00] and LCAD-UFS for providing a cluster for the execution of the experiments. The authors also thank FAPITEC-SE for granting a graduate scholarship to Flávio Santos, and CNPq for giving a productivity scholarship to Hendrik Macedo [DT-II, Processo 310446/2014-7].

References

[1] Abdulrahman Almuhareb and Massimo Poesio. Concept learning and categorization from the web. In *proceedings of the annual meeting of the Cognitive Science society*, volume 27, 2005.

[2] Oded Avraham and Yoav Goldberg. The interplay of semantics and morphology in word embeddings. *arXiv preprint arXiv:1704.01938*, 2017.

[3] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee, 2017.

[4] Marco Baroni and Alessandro Lenci. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10. Association for Computational Linguistics, 2011.

[5] William F Battig and William E Montague. Category norms of verbal items in 56 categories a replication and extension of the connecticut category norms. *Journal of experimental Psychology*, 80(3p2):1, 1969.

[6] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[7] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.

[8] Elia Bruni, Nam-Khanh Tran, and Marco Baroni. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47, 2014.

[9] Alexander Budanitsky and Graeme Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and other lexical resources*, volume 2, pages 2–2, 2001.

[10] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

[11] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[12] Ryan Cotterell and Hinrich Schütze. Morphological word embeddings. *arXiv preprint arXiv:1907.02423*, 2019.

[13] Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 142–151, 2017.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM, 2001.

[16] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[17] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695, 2015.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] Stanisław Jastrzebski, Damian Leśniak, and Wojciech Marian Czarnecki. How to evaluate word embeddings? on importance of data efficiency and simple supervised tasks. *arXiv preprint arXiv:1702.02170*, 2017.

[20] David A Jurgens, Peter D Turney, Saif M Mohammad, and Keith J Holyoak. Semeval-2012 task 2: Measuring degrees of relational similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 356–364. Association for Computational Linguistics, 2012.

[21] Carlos AEM Júnior, Luciano A Barbosa, Hendrik T Macedo, and SE São Cristóvão. Uma arquitetura híbrida lstm-cnn para reconhecimento de entidades nomeadas em textos naturais em língua portuguesa. *XIII Encontro Nacional de Intelig^encia Artificial e Computaciona*, 2016.

[22] Himabindu Lakkaraju, Richard Socher, and Chris Manning. Aspect specific sentiment analysis using hierarchical deep learning. In *NIPS Workshop on Deep Learning and Representation Learning*, 2014.

[23] Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. *arXiv preprint arXiv:1612.01887*, 2016.

[24] Thang Luong, Richard Socher, and Christopher Manning. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, 2013.

[25] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[27] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

[28] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[29] Marco Aurelio Moura. *O Discurso do Ódio em Redes Sociais*. Lura Editorial (Lura Editoração Eletrônica LTDA-ME), 2016.

[30] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[31] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[32] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[33] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[34] Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. Co-learning of word representations and morpheme representations. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 141–150, 2014.

[35] Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 337–346, New York, NY, USA, 2011. ACM.

[36] Herbert Rubenstein and John B. Goodenough. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, October 1965.

[37] Rana Aref Salama, Abdou Youssef, and Aly Fahmy. Morphological word embedding for arabic. *Procedia computer science*, 142:83–93, 2018.

[38] Flávio Arthur O. Santos and Hendrik T. Macedo. Improving word representations using paraphrase dataset. In Shahram Latifi, editor, *Information Technology - New Generations*, pages 405–409, Cham, 2018. Springer International Publishing.

[39] Peter Smit, Sami Virpioja, Stig-Arne Grönroos, Mikko Kurimo, et al. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *The 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Gothenburg, Sweden, April 26-30, 2014*. Aalto University, 2014.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[41] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[42] Yang Xu and Jiawei Liu. Implicitly incorporating morphological information into word embedding. *arXiv preprint arXiv:1701.02481*, 2017.