Fake News Detection Using Recurrent Neural Networks and Distributed Representations for the Portuguese Language

Guilherme Zanini Moreira[1], Marcelo Romero[2], Manassés Ribeiro[1]
[1]Catarinense Federal Institute of Education, Science and Technology, Videira, Brazil
[2]Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France
gzaninimoreira@gmail.com, nmarceloromero@gmail.com, manasses.ribeiro@ifc.edu.br

***Abstract*—After the advent of Web, the number of people who abandoned traditional media channels and started receiving news only through social media has increased. However, this caused an increase of the spread of fake news due to the ease of sharing information. The consequences are various, with one of the main ones being the possible attempts to manipulate public opinion for elections or promotion of movements that can damage rule of law or the institutions that represent it. The objective of this work is to perform fake news detection using Distributed Representations and Recurrent Neural Networks (RNNs). Although fake news detection using RNNs has been already explored in the literature, there is little research on the processing of texts in Portuguese language, which is the focus of this work. For this purpose, distributed representations from texts are generated with three different algorithms (fastText, GloVe and word2vec) and used as input features for a Long Short-term Memory Network (LSTM). The approach is evaluated using a publicly available labelled news dataset. The proposed approach shows promising results for all the three distributed representation methods for feature extraction, with the combination word2vec+LSTM providing the best results. The results of the proposed approach shows a better classification performance when compared to simple architectures, while similar results are obtained when the approach is compared to deeper architectures or more complex methods.**

***Index Terms*—Fake news detection; Word embedding; Recurrent neural networks; Long short-term memory**

I. Introduction

Fake news exist even before the emergence of the Internet. Generally, fake news imply that editors use false or misleading information to promote their interests. After the advent of web, more consumers began to abandon traditional media channels, thus producing a growing segment of population receiving news only through social media [1], [2]. The term "fake news" can be applied when information is published with no guarantee of being true, its credibility cannot be proven, or when false information is spread for the purpose of deceiving people [3].

The degree of dissemination of false information that is present in social media has significantly expanded during the recent years, due to the ease of sharing information. Moreover, there is a growing presence of *bots*[1] that manipulate the social media recommendation algorithms by interacting with fake news posts. The main purpose of false news is monetising through advertisements. However, there are claims that even presidential elections in important countries may have been manipulated through the use of false news. Detecting this type of news is not only relevant for the society, but also for the technology and social media companies such as *Google*, *Facebook* and *Twitter* [1], since it implies the responsibility of the social network (and the company) to avoid mass disinformation to its users. We have seen this emerging concern in combating misinformation during the coronavirus (COVID-19) pandemic. Thus, initiatives have been proposed throughout the years to identify fake news using both automated and manual checking methods.

The existence of tools to support automatic detection of fake news in languages other than English, more specifically in Portuguese, is still unsubstantial in both, the industry and academia. This is due to that the computational methods for this purpose are mostly developed for English-language texts. However, some initiatives to detect fake news for the Portuguese language have been proposed in an attempt to reduce their spread. One example is Facebook, which financed the development of chat-bot for the Portuguese language to help users learn how to check the veracity of photos, videos, rumours, news and false statements[2].

Note that the task of identifying fake news among a set of news is a classification problem, which is tackled through algorithms that assign labels to samples, considering a set of characteristics. In the specific context of fake news classification, it is necessary to use text-oriented feature extraction methods and classification algorithms that are able to process the temporal dependencies that appear in sentences and long texts. Regarding text-oriented feature extraction, various Natural Language Processing (NLP) techniques that are able to capture similarity relationships between words have been introduced during the recent years, namely Word2vec [4], GloVe [5] e fastText [6]. These methods are based on the use of artificial neural networks to obtain dense word representations. On the other hand, considering classification algorithms, Deep Learning (DL) methods have substantially improved the state-of-the-art in several domains, such as speech recognition [7]

[1]Automated accounts representing people with the aim of influencing the course of the discussion

[2]https://www.aosfatos.org/fatima/

and object detection [8], [9]. These methods allow computational models composed of several layers of abstraction to learn data representations [10]. A widely used DL method for processing sequential data such as text and speech is the Recurrent Neural Network (RNN) [11], and architectures derived from it such as the Gated Recurrent Unit (GRU) [12] Long Short-Term Memory (LSTM) [13].

This work proposes a computational approach for the classification of fake news, written in Portuguese, using a LSTM-type recurrent neural networks and distributed representations. Specifically, this work aims at accomplishing the following objectives: (1) Evaluate the main methods available in the distributed representation literature (*word embedding*); (2) Compare the classification performance of the chosen *word embedding* methods; (3) Verify the use of LSTM-type RNNs for text classification, as well as their classification performance for the fake news classification in the Portuguese language; and (4) Compare the proposed approach to other state-of-the-art approaches to text classification.

In summary, the main contributions of this work are: (1) experimentation of the main approaches of distributed representations for text classification (word2vec, GloVe and fastText), thus comparing their classification performance; (2) proposal of an approach for the purpose of classification of false news for Portuguese language; (3) Use of LSTM-type RNNs associated with distributed representation methods for detecting false news, by training our approach using a novel labelled text corpus.

This work is organised as follows: Section II presents the theoretical aspects of this work. Section III describes the related work. Section IV presents the proposed approach. Section V details the experiments, results and a brief discussion around them. Finally, Section VI presents the final considerations of this paper and describes possible future research paths.

II. Background

This section presents relevant background aspects for understanding the method that is proposed in this work. Specifically, this section introduces concepts such as distributed representation and LSTM networks.

A. Word Embedding

Word embedding is a set of techniques that derive from the distributive hypothesis of language, which assumes that the meaning of a word can be inferred from the contexts in which that word is used [14], and in the context of this work, it is used as distributed representation.

Methods based on *embeddings* represent each word as dense vectors of real values with predefined dimensions. According to [15], vectors generated from *word embedding* may be used as input for different tasks of NLP, such as natural language recognition, similarity between words, information retrieval, document classification and sentiment analysis.

A *word embedding* is computed by applying dimensionality reduction to the co-occurrence matrix that is produced from a large set of texts, known as *corpus* [15]. A co-occurrence matrix consists of distributional vectors containing values found in cells of a row [16]. For instance, the vector of *oil-ship* in the co-occurrence matrix presented in Table I is $X_{oilship}$ = (67, 62, 83, ...). The content of each position in the vector represents the value that it has in the co-occurrence matrix (number of times the oil tanker co-occurs with the ship, for example).

TABLE I
Co-occurrence matrix (hypothetical), adapted from [16]

	Co-occurring words			
	to load	load	ship	⋯
crew	54	58	150	⋯
oil	61	58	85	⋯
refinery	50	80	80	⋯
sea	4	10	100	⋯
oil tanker	67	62	83	⋯

Therefore, similar terms tend to be positioned in the same neighbourhood region in the vector space, thus making it possible to capture essential language features such as syntax and semantics. In this case, similarity relationships may be inferred from simple measures of distance and similarity between vectors. The most popular word vectorisation approaches are Word2vec, *Global Vectors* (GloVe) and fastText [17].

1) Word2vec: Word2vec is a model proposed by [4] that aims at obtaining ANN-based word vector representations [18]. Word2vec uses an input layer of dimension v, which takes a vector in *one-hot* representation as input; a hidden layer of dimension n, where n refers to the dimension of the vectors that will represent the words; and an output layer of dimension v, which returns vectors similar to the *one-hot* representation, but instead of 0's and 1, these vectors express probabilities for each vocabulary word [19].

The *one-hot* representation is the length of the vocabulary size and it is filled with zeros, except for the index representing the word that is being represented, which is filled with 1. The hidden layer is a fully-connected layer, and its weights are, in fact, the *embedding* of the word. The output layer generates probabilities for the vocabulary target words [20]. The synapses that connect the input layer to the hidden layer, and the hidden layer to the output layer, can be represented by the matrices $\mathbf{WI}_{v\times n}$ and $\mathbf{WO}_{n\times v}$, respectively [19].

The operation $r = (\mathbf{x} \times \mathbf{WI}) \times \mathbf{WO}$ represents the computation of the vector $\mathbf{x}$ from input layer to hidden layer and then from hidden layer to output layer, the purpose of which is to learn hidden layer weights. Finally, Word2vec is trained using a backpropagation approach and transforms the output layer activation values into probabilities through the *softmax* function.

Word2vec proposes two different training architectures: Continuous Bag-of-Words (CBOW) and *Skip-gram* [17]. In the CBOW architecture, a word $\mathbf{w}_i$ is predicted based on its context, which consists of a word window composed of terms before and after $\mathbf{w}_i$ [19]. The *skip-gram* architecture,in turn, seeks to maximise the context prediction based on a word $\mathbf{w}_i$ given as input. The main difference between the architectures is that in CBOW the input layer is replicated C times, where C is the amount of context words, while in the *Skip-gram* it is the output layer that goes through this replication [19].

2) GloVe: GloVe is a model proposed by [5], and its difference from the two architectures proposed by Word2vec is that it is a model based on word counts. In GloVe, the co-occurrence matrix is used to generate *embedding*, that is, how often a specific term appears among another in a *corpus* [21]. The notation used by GloVe to calculate the probability of a word j occurring in the context of the word i is defined by the Equation 1.

$$P_{i,j} = \frac{\mathbf{x}_{i,j}}{\sum_{k=0}^{n} \mathbf{x}_{i,k}}, \tag{1}$$

where $\mathbf{x}$ denotes a vector of co-occurrence, the input $\mathbf{x}_{i,j}$ displays the number of times the word j appeared in the context of the word i and n the number of words that co-occur with the word i.

In the first instance the *embedding* of the words are generated randomly, as at the Word2vec. In this sense, [5] proposes a function to perform adjustments in the *embedding* such that the dot product of their vectors are similar to their values in the co-occurrence table. The function that performs the adjustments is presented in Equation 2.

$$F(\mathbf{w}_i^T \tilde{\mathbf{w}}_k) = P_{ik} = \frac{\mathbf{X_{ik}}}{\mathbf{X}_i}, \tag{2}$$

where $\mathbf{w}_i^T$ is the transposed vector of $\mathbf{w}_i$, $\tilde{\mathbf{w}}_k$ one of the *embedding* vectors of the word, P_{ik} the ratio of the word i with the context word k and $\mathbf{X}$ is the co-occurrence matrix.

As the F function is not symmetric, a suitable F' function is defined in Equation 3, in order to guarantee symmetry [21].

$$F'(\mathbf{w}_i, \mathbf{w}_k) = \mathbf{w}_i^T \tilde{\mathbf{w}}_j + b_i + \tilde{b}_i \tag{3}$$

where b_i and $\tilde{b}_i$ are the *bias* for the words $\mathbf{w}_i$ and $\mathbf{w}_k$, respectively. $\mathbf{w}_i^T$ and $\tilde{\mathbf{w}}_j$ are the vectors of *embedding*.

Finally, a summation covering the entire V vocabulary is used to generate the *embedding*.

3) fastText: fastText is a model developed by a team of researchers from *Facebook* [6], as an extension of the Word2vec model, where the goal is to learn vectors and perform classification of text [22].

For word representation, fastText is based on the *Skip-gram* model. The algorithm differs from Word2vec in that it assumes that words are composed of *n-grams* of their characters. The algorithm learns the vector representations distributed for the *n-grams* of vocabulary words, and forms the distributed vector representation of a word as the sum of the representations of its *n-grams* [22]. However, the training process takes longer compared to other [23] models.

fastText works by sliding a window over the input text and learning all context words from the central word. The difference between fastText and the *Skip-gram* architecture is in the probability function, where it is changed so that the *n-grams* are counted [23].

B. Long Short-Term Memory

LSTM is an RNN architecture designed to model long-term temporal sequences more accurately than conventional RNN [24]. LSTM was introduced by [13] with the motivation of offering better performance by solving the vanishing problem that RNNs naturally suffer when dealing with large data streams [25].

According to [10], the main difference between LSTM and traditional RNNs is that the LSTM architecture has more parameters and also a strategy of information flow control. This strategy is present in the LSTM hidden layer, and it consists of special units called memory blocks, which contain two basic structures: cells that have connections to other temporal storage cells and weighted units called *gates* that to control the information flow [24]. Figure 1 shows a general diagram of the LSTM, where it presents the information flow of the cell C_t and the strategy to control information flow. The operation represented by X at the Figure correspond to the Hadamard product, which produces a new weighted matrix with same dimension [26].

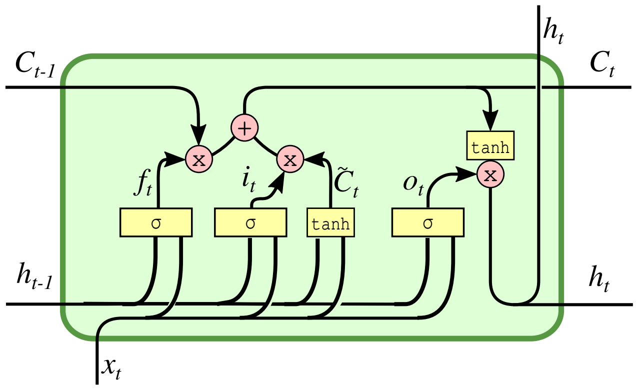

Fig. 1. LSTM architecture adapted from [27].

According to [28], the LSTM is divided into three main *gates*:

- *Forget*: the information flow of the LSTM starts at the forget gate (f_t), which decides what information will remain at the cell state C_t. Two inputs, x_t and h_{t-1}, are fed to the gate and multiplied by weight matrices. Their resultant is made by assigning a value between $[0, 1]$, where the output 0 is forgotten and for output 1 the information is retained for future use.
- *Input*: Useful information addition to the cell state is done by the input gate. First, the information is regulated using the Sigmoid function which filters the values to be remembered similarly to forget gate using the h_{t-1} and x_t inputs. Then, a vector is created using the hyperbolic

tangent function that outputs $[-1, 1]$, which contains all possible values of h_{t-1} and x_t. Vector values and calibrated values are multiplied to get useful information.
- *Output*: the output gate is responsible for the task of extracting useful information from the current cell state to be presented as an output. First, a vector is generated by applying the hyperbolic tangent function to the cell. Then, the information is regulated using the Sigmoid function that filters the values to be remembered using the inputs h_{t-1} and x_t. Vector values and regulated values are multiplied to be sent as an output and input to the next cell.

III. Related Work

The work by [29] proposed the classification of fake news extracted from Brazilian websites using text summarisation techniques, which consist of reducing large textual parts into smaller ones. In this approach, the word2vec word embedding method with dimensionality of 10 was used for feature extraction. For the classifier, the author proposed the use of two Deep Learning (DL) algorithms and two traditional artificial intelligence algorithms. The DL algorithms used in the work were Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM), while the traditional artificial intelligence algorithms were: Random Forests and Support Vector Machines (SVM). The author reported that the experiments performed with the LSTM neural network achieved a higher accuracy, around 79.3%. Despite getting a result close to 80.00%, it is possible to evaluate that the dimensionality of the embedding vector of the words being 10 can be small, considering that a DL algorithm was used. Another relevant factor for obtaining this result could be the size of the database used in the work, in which 286 news were used for each class.

The work proposed by [30] presented predictive models to classify fake news using word embedding techniques combined with Machine Learning (ML) and DL algorithms. The word embedding methods chosen for feature extraction were GloVe and word2vec. The version of GloVe used in the work was the pre-trained word embedding with 300 dimensional feature vector, trained with corpus extracted from *Wikipedia* and *Gigaword* 5. The ML algorithms used in the work were: Logistic Regression[3], Random Forests, Naive Bayes, SVM, and xgboost. The DL algorithms were: feed-forward and convolutional neural networks. The best model was the one that combined convolutional neural networks and GloVe with an accuracy of a value around 97.50%. The second best accuracy, around 91.37%, was achieved by feed-forward neural network combined with word2vec. Although it is not possible to evaluate the Word Embedding techniques because the embeddings of the words were trained using different datasets, it is possible to evaluate that the model with the embeddings trained using another dataset presented the best result.

[3]Algorithm used to predict the probability of a sample belonging to a class by learning how each characteristic correlates with the final result.

The classification of fake news using the Hierarchical Attention Networks (HAN) architecture and word2vec is proposed by [31]. HAN allows to visualise the output data through a heat map that gives insight regarding the reason why a certain class is chosen. It highlights the words and sentences that considered most important for the classification by the network. The approach achieved an accuracy value close to 95.35%. On the other hand, [15] proposed a comparative study of word embedding methods for sentiment analysis. The work used a neural network as a classifier and word embeddings for feature extraction. The word embedding methods used in the work were Latent Semantic Anaylsis, word2vec, GloVe and Sentiment-Specific Word Embedding (SSWE), with feature vectors of dimensions equal to 50, 100 and 300. The objective of a sentiment analysis classification model is to classify the polarity of the review of a movie. The dataset used for training and testing the models is the *Internet Movie Database*. The results of the experiments showed that the best accuracy was obtained when SSWE was used, achieving an accuracy equal to 82.12%.

The work proposed by [32] addresses the classification of fake news using two different classification methods: LSTM and Logistic Regression. The approach using LSTM had as input a word embedding vector obtained through other corpus in Portuguese provided by [33]. On the other hand, the Logistic Regression algorithm used as input a statistical representation. The best result was achieved by the Logistic Regression approach, with an *f1-score* of 92.8%, whilst the approach based on the use of LSTM and GloVe with a dimensionality equal to 1000 obtained a *f1-score* of 89.6%.

The objective of this work is to detect false news using distributed representations and LSTM. For this purpose, the methods fastText, GloVe and word2vec are used in order to obtain distributed representations of words. We train and test our approach using a novel labelled text corpus.

IV. Methodology

This section presents the methods proposed to tackle the problem, which is the detection of fake news in texts. We propose methods considering the use of Deep Learning (DL) and text-oriented feature extraction, which have been widely used for text classification. The main hypothesis that we intend to explore in this work addresses the possibility that fake news can be identified through a computational model composed of distributed representations and a kind of Recurrent Neural Networks (RNNs), named *Long Short-Term Memory* (LSTM) architecture. The LSTM architecture has been selected due to its capability to process long sequences of data and given that it allowed to achieve state-of-art results for various tasks related to text processing. Distributed representations are obtained using techniques known as *word embeddings*, which allow better generalisation for classification models due to similarity relationships between words.

Figure 2 presents the overview of the proposed method. During the first stage, the dataset is obtained and the data preprocessing is performed, thus selecting only relevant words, standardising all words to lowercase, and removing all spelling accents. After this stage, the distributed representations are obtained through a word embedding process using the following methods: Word2vec, GloVe and fastText. In the training stage, the LSTM architecture is built and training samples are fed to the model in a sequence of epochs until reaching a convergence phase. After the training stage, the test samples are fed to the model in order to evaluate its classification performance. Finally, the model performance is verified through metrics and according to the classification that was obtained from the test phase.

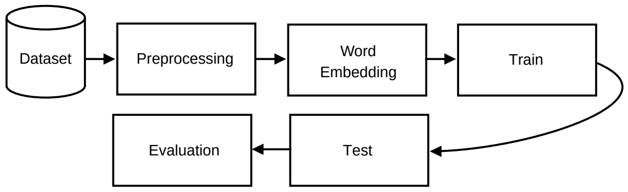

Fig. 2. Overview of the proposed method.

A. Data Preparation

1) Pre-processing: Pre-processing is necessary to standardise the text and select features, removing words that may be considered irrelevant (known as *stop-words*). First, a word tokenization method is used, which is a method that divides a large sample of text into words. For this, the natural language toolkit NLTK library[4] is used.

After, as mentioned above, all letters of all words are converted from uppercase to lowercase, and then all graphic accents and diacritics are removed [5], as well as all words considered irrelevant and all non-alphabetic characters. The dataset resulting from the pre-processing stage will be used as input to the word embedding stage (distributed representation) with the three mentioned methods (Word2vec, GloVe and fastText).

2) Distributed representations: After the pre-processing process, it is necessary to generate the distributed representations. As mentioned before, this is achieved using *word embedding* methods, which allow to generate vectors for each word that will afterwards be used as input to the LSTM model. For this procedure, two different libraries are used: the *gensim*[6] for both Word2vec and fastText methods, and the GloVe[7].

The use of *gensim* (for both Word2vec and fastText) requires two steps. First, the pre-processed text is loaded and the sentences are converted into a vector, where each element of the vector is a text-word. The second step is the proper training of each model using parameters presented in Table II, with the vector containing the text as input. For generating the *word embedding* using the GloVe library, it is necessary to create a *script bash* with the parameters shown in Table II, including the directory where the text files are. Finally, the word and its *embedding* will compose both training and test datasets.

Table II shows each parameter necessary to generate the *word embedding*. Values shown in the *Size* parameter were chosen according to the sizes of a standard *word embedding* values, which was obtained from training with large text corpus, for both English and Portuguese languages, and they are publicly available. Other parameters were set according to the values presented in [22].

TABLE II
PARAMETERS FOR TRAINING WORD EMBEDDING

Parameter	Description
size: 50, 100 e 300	Vector sizes (dimension)
window: 5	Maximum distance between current word and predicted word
min_count: 0	Ignore all words with frequency less than
alpha: 0.05	Learning rate
iter: 10	Number of interactions (epochs) on the *corpus*

3) Word dictionary: At this step, a word dictionary is created, assigning an identification number to each word. Here, the text of each news is converted into a list of integers, representing the identifier of each word. After this step, all news' text is transformed into a one-size-fits-all string, according to the average of words of the entire dataset. This way, texts with the number of words smaller than the defined length will be filled with 0's. This step of transforming all texts to a single size is necessary to train the model in batches.

4) Dataset division: Finally, we use the holdout method to divide the dataset into two subsets at random. The first one is used just for training (to adjust the LSTM model), whilst the second one is used for testing the model classification performance. The split ratio is 80% for training, and 20% for testing.

B. Model architecture

During this step, the model architecture is built. The model is an Artificial Neural Network composed of three layers: an input layer (embedding) that has the task of loading the pre-trained vector of each news word, a LSTM layer with output dimensionality equal to the dimensionality size of the input vector (50, 100 and 300), and a dense layer with output size 2, with the softmax activation function. The loss function chosen is the *binary cross-entropy* due to that the classification is binary, and the chosen optimiser is *RMSProp*. It is worth mentioning that our model is inspired on the work proposed by [34]. Figure 3 illustrates the proposed model, with a news story composed of 5 words. For both input layer (embedding) and LSTM representation, the dimensionality is 4, and the dense layer is the model output with size 2.

C. Training and Test

At this stage, the samples from the training subset are crossed through the model for n training epochs, until the

[4]www.nltk.org/

[5]Symbols that are added to the letters of the alphabet to indicate the different pronunciation that the letters usually receive.

[6]www.radimrehurek.com/gensim

[7]www.github.com/stanfordnlp/GloVe

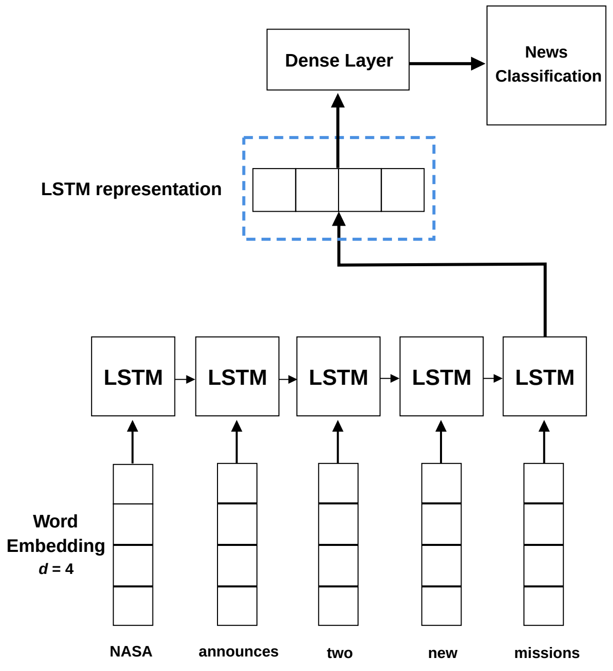

Fig. 3. Illustration of model architecture.

model reaches the appropriate level of convergence. Afterwards, the model is tested with samples from the test subset only.

D. Evaluation

To evaluate the proposed model, we use the confusion matrix, whose indicators are:

- TP: the prediction is as fake news and text is about fake news.
- FP: the news is predicted as fake news, but the text is about a true news.
- TN: the prediction is as true news and the text is true news too.
- FN: the news is predicted as true news, but the text is fake news.

The metrics are computed using the confusion matrix indicators (precision, recall, f1-score and accuracy) . These are used to evaluate the performance of the model. Besides, the mentioned metrics are be used to evaluate the different word embedding methods for the classification, as well as to compare their results with those from other works.

V. Experiments and Results

This section presents the experiments that were performed as well as the results achieved with the methodology described in Section IV. The hypothesis of this work is that it is possible to obtain a fake news classification model through the use of distributed representations and RNNs for a text corpus defined in Portuguese. The experiments were conducted with the objective to test this hypothesis. We also present the dataset that was used to perform the experiments. Notice that all experiments were run in a dedicated GPU server with an Intel i7-5820K CPU running at 3.3 GHz, with 32 GB of RAM and equipped with a Nvidia Titan-Xp GPU, running on Ubuntu.

A. Dataset

The experiments that were performed in this work were conducted using the *Fake.Br* [35] dataset. This dataset contains $7,200$ news, with $3,600$ being fake news and $3,600$ being true news. The news were analysed and extracted from news websites, with the publication period being between January 2016 and January 2018.

The set of news composed of fake news from the *Fake.Br* corpus was collected manually, while the news that form part of the true news sub-dataset were collected in a semi-automatic manner. According to 35, the first stage of the collection was performed using a web crawler[8], seeking news based on keywords from a fake news story. After this step, for each fake news, a measure of lexical similarity with the real news collected was applied, choosing the most similar to the fake news. The last step of the collection of the true news was a manual check to ensure that for every true news there would be a fake one related to the same subject.

The dataset can be broken down into 6 categories related to their main subjects: politics, TV and celebrities, society and daily news, science and technology, economics and religion. The amount of news per category in *corpus Fake.Br* is as follows: $4,180$ political news, $1,544$ TV and celebrity news, $1,276$ society news and daily news, 112 science and technology news, 44 economy news and 44 religion news.

The average number of words for each class is presented in Figure 4. Moreover, the average number of words in the text after the pre-processing step is also shown. Figure 4 shows the relevance of the pre-processing step in order to reduce the size of news by removing words that can be considered irrelevant, which allows to reduce the amount of inputs for the classification model.

B. Results and discussion

The objective of this study is to verify the classification performance of the model using the representation generated through fastText, GloVe, and word2vec; as well as verifying the performance of the architecture using LSTM. All classification performance metrics used in this work are computed from the confusion matrix.

Model training was done with the $7,200$ news dataset, with a split of 80% for training and 20% for testing. The dataset split process was performed 10 times, so each method was performed 10 times with different datasets and testing.

The means and standard deviations of the performance metrics obtained through the confusion matrices can be seen in Table III, where the three studied methods of word embeddings and their dimensions are presented.

[8]Robot to extract data from pages on the Internet

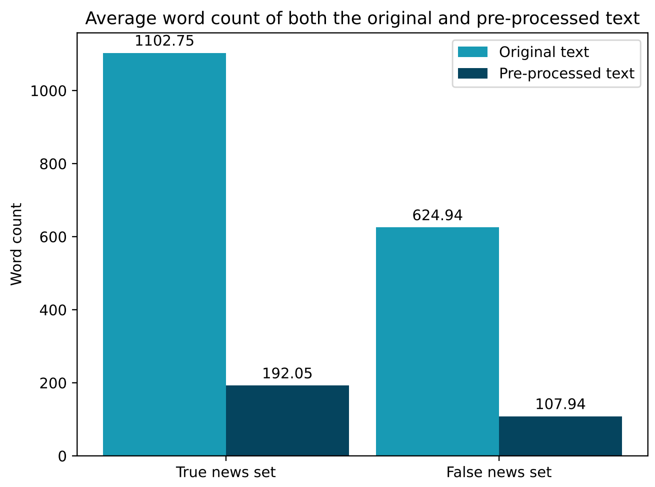

Fig. 4. Average word count of both the original and pre-processed text for the true and false news set.

TABLE III
PERFORMANCES ACHIEVED BY THE CLASSIFIER USING THE DIFFERENT WORD EMBEDDINGS AND DIMENSIONS FOR THE FEATURE VECTORS.

	fastText			**GloVe**			**word2vec**		
Dimension	50	100	300	50	100	300	50	100	300
Precision	**0.9555 ±0.0056**	0.9515 ±0.0116	0.9450 ±0.0047	**0.9528 ±0.0054**	0.9499 ±0.0082	0.9462 ±0.0106	0.9546 ±0.0046	0.9553 ±0.0072	**0.9573 ±0.0036**
Recall	**0.9551 ±0.0060**	0.9500 ±0.0150	0.9449 ±0.0046	**0.9524 ±0.0061**	0.9488 ±0.0096	0.9461 ±0.0106	0.9542 ±0.0048	0.9551 ±0.0074	**0.9571 ±0.0037**
f1-score	**0.9551 ±0.0061**	0.9499 ±0.0152	0.9449 ±0.0046	**0.9524 ±0.0061**	0.9488 ±0.0097	0.9461 ±0.0106	0.9542 ±0.0048	0.9551 ±0.0074	**0.9571 ±0.0037**
Accuracy	**0.9551 ±0.0060**	0.9500 ±0.0150	0.9449 ±0.0046	**0.9524 ±0.0061**	0.9488 ±0.0096	0.9461 ±0.0106	0.9542 ±0.0048	0.9551 ±0.0074	**0.9571 ±0.0037**

The results presented in Table III shows that the increase in the size of embeddings and consequently of the news resulted in a performance improvement only for word2vec. For fastText and GloVe, the best results were obtained using embeddings of dimension size equal to 50. Therefore, the model failed to extract relevant features with a higher dimensionality.

The model that used vectors with dimension size equal to 300 generated using the word2vec achieved the best results, since this method generates embeddings based on words in similar contexts. The performance of the model that used fastText achieved the second best result.

The results obtained for all embedding methods showed satisfactory absorption of the specific context, contributing to an overall average of 95.44% accuracy for the model and an average value of 0.9515 for the *f1-score* metric, thus making it possible to verify that there is a balance between precision and recall.

Table IV, shows a comparison of the results obtained with the model proposed in this work and the results of other works aimed at detecting false news. The work of [31] uses Hierarchical Attention Networks (HAN) and the word2vec with dimensionality equal to 100. The work of [32] addressed two different ML algorithms and two methods of document representation, and the approach using LSTM has as input embeddings generated with GloVe, whereas the Logistic Regression algorithm inputs data obtained through statistical representations. Therefore, only the approach that uses word embeddings is compared to our approach. On the other hand, the work by [30] uses feed-forward neural networks and GloVe with dimensionality equal to 300 and a dataset with $27,985$ news in the English language. The dataset used by [31] and [32] is the *Fake.br* corpus, whereas the embeddings used in these works are pre-trained vectors with other corpus in Portuguese and publicly available in [33].

TABLE IV
PERFORMANCES ACHIEVED BY THE CLASSIFIER

Method	Word Embedding	Precision	Recall	f1-score	Accuracy
HAN [31]	*word2vec* - 100	0.9534	0.9538	0.9535	0.9535
Feed-forward [30]	*GloVe* - 300	0.8759	0.9231	0.8989	0.9126
LSTM [32]	*GloVe* - 1000	0.8860	0.9080	0.8970	0.8960
LSTM	*fastText* - 50	0.9555	0.9551	0.9551	0.9551
LSTM	*fastText* - 100	0.9515	0.9500	0.9499	0.9500
LSTM	*fastText* - 300	0.9450	0.9449	0.9449	0.9449
LSTM	*GloVe* - 50	0.9528	0.9524	0.9524	0.9524
LSTM	*GloVe* - 100	0.9499	0.9488	0.9488	0.9488
LSTM	*GloVe* - 300	0.9462	0.9461	0.9461	0.9461
LSTM	*word2vec* - 50	0.9546	0.9542	0.9542	0.9542
LSTM	*word2vec* - 100	0.9553	0.9551	0.9551	0.9551
LSTM	*word2vec* - 300	0.9573	0.9571	0.9571	0.9571

From the results it is possible to assess that there is no significant improvement from the proposed model when compared to the approach that relies on HAN [31]. However, since the approaches proposed by [31] and [32] use pre-trained vectors in the same dataset, it is possible to verify that the approach proposed in this work obtained a better classification performance compared to the work of [32], which uses the same dataset and ML algorithm.This better performance can be explained by the fact that the vectors were pre-trained using the same dataset that was used for testing. However, it is possible to verify that approaches that use LSTM-type RNNs tend to obtain satisfactory results for classifying texts using word embeddings as feature representations.

In comparison with simpler architectures such as feed-forward neural networks, proposed by [30], our approach achieved better classification performance, despite that the dataset used in our approach contains $7,200$ news and the dataset used by [30] contains $27,985$.

VI. Conclusion

Due to the growing concern with impact that fake news may cause to the society, it emerges the need for automated methods for detecting fake news using computational approaches. Methods such as DL and NLP have been widely used to assign categories to texts according their content. In this sense, distributed representations of words and RNNs are widely used for text classification tasks. Thus, in order to contribute to the detection of fake news in texts, this work focused on obtaining distributed representation from words with three methods of *word embedding*, and thus classifying news using type-LSTM RNNs.

At the experiment conducted to verify the classification performance concern *word embedding* methods, it was observed that these methods are capable of capturing context-based characteristics. The best classification performance was obtained using the vectors generated by the Word2vec method, however, the results obtained with the other studied methods were similar. At this sense, results suggest that the proposed approach is promising and may be used as a tool for ranking fake news. Regarding the results obtained from conducted experiment using *embedding layer* with a dimensionality of 300, these were better when compared with the dimensions 50 and 100, just for the Word2vec.

In comparison of our architecture with different others proposed in literature it is possible to verify similar results, despite others are more deep (and more sophisticated) architectures, obtaining good classification performance with average value about 0.9515 to the *f1-score* metric. The architecture proposed at this work seems to be better in classification performance when compared to simpler architectures.

For training and test step, a publicly available dataset containing true and fake news in Portuguese was used. From obtained results it is possible to evaluate that the suggested approach using *word embedding* methods together with LSTM is capable of classifying fake news at Portuguese language story. Overall, from results, we believe that our approach is very promising for detecting fake news on news stories, and it could be generalised to other news corpus.

As future works, a possible research direction is to apply new methods for processing natural language, such as encoder architectures based on *Transformer* DL model such as the *Bidirectional Encoder Representations from Transformers* (BERT). BERT is a model already trained in a large text base and which can be retrained on new data (like as the problem in question) in order to learn the difference between the general meaning of a word and the meaning of the word for a specific context. This issue is specially important for fake news classification, because the context understanding is a not trivial task and the main objective for a good classification of fake news is to minimise the semantic gap between general meaning and specific meaning of words. Another possible direction of research is the detection of fake news from text on images, thus using a optical character recognition early.

Acknowledgement

All authors thanks NVIDIA for the donation of Titan-Xp GPU boards used in this work.

References

[1] M. Aldwairi and A. Alwahedi, "Detecting fake news in social media networks," *Procedia Computer Science*, vol. 141, pp. 215 – 222, 2018.

[2] A. Figueira and L. Oliveira, "The current state of fake news: challenges and opportunities," *Procedia Computer Science*, vol. 121, pp. 817–825, 12 2017.

[3] D. M. J. Lazer *et al.*, "The science of fake news," *Science*, vol. 359, no. 6380, pp. 1094–1096, 2018.

[4] T. Mikolov *et al.*, "Efficient estimation of word representations in vector space." Mountain View, CA, EUA: Google, 01 2013, pp. 1–12.

[5] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. Palo Alto, CA, EUA: Stanford University, 2014, pp. 1532–1543.

[6] P. Bojanowski *et al.*, "Enriching word vectors with subword information," *CoRR*, vol. abs/1607.04606, 2016.

[7] G. Saon *et al.*, "English conversational telephone speech recognition by humans and machines," *CoRR*, vol. abs/1703.02136, 2017.

[8] A. Kolesnikov *et al.*, "Big transfer (bit): General visual representation learning," 2019.

[9] H. Touvron *et al.*, "Fixing the train-test resolution discrepancy: Fixefficientnet," 2020.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, MA, USA: MIT press, 2016.

[11] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, pp. 2554–8, 05 1982.

[12] J. Chung *et al.*, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint*, 2014.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 17351780, Nov. 1997.

[14] Y. Goldberg, "Neural network methods for natural language processing," *Synthesis lectures on human language technologies*, vol. 10, no. 1, pp. 1–309, 2017.

[15] M. H. d. Carvalho, "Estudo comparativo dos métodos de word embedding na análise de sentimentos," Recife, Pernambuco, Brasil, 2018.

[16] B. H. Al Farsi, "Word meaning in word identification during reading: Co-occurrence-based semantic neighborhood density effects," *Applied Psycholinguistics*, vol. 39, no. 5, p. 779809, 2018.

[17] D. Gomes and A. Evsukoff, "Processamento de linguagem natural em português e aprendizagem profunda para o domínio de óleo e gás," 2019.

[18] R. Aguiar and R. Prati, "Incorporao de representao vetorial distribuda de palavras e pargrafos na classificao de sms spam." Santo Andr, SP, Brasil: Universidade Federal do ABC (UFABC), 11 2015.

[19] R. d. A. Lima, "Classificação de polaridade de sentimentos de mensagens curtas utilizando word2vec," 2017.

[20] J. Gilyadov, "Word2vec explained," 2017.

[21] S. d. Sousa, "Estudo de modelos de word embedding," B.S. thesis, Universidade Tecnológica Federal do Paraná, 2016.

[22] J. T. Pastro, "Detecção de tweets sobre eventos de trânsito usando word embedding," 2018.

[23] I. P. P. dos Santos, "Análise de sentimento usando redes neurais de convolução," Ph.D. dissertation, Universidade do Estado do Estado do Rio de Janeiro (UERJ), Rio de Janeiro, Rio de Janeiro, Brasil, 2017.

[24] H. Sak, A. W. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *CoRR*, vol. abs/1402.1128, 2014.

[25] D. M. Q. Nelson, "Uso de redes neurais recorrentes para previsão de séries temporais financeiras," 2017.

[26] É. Renault, P. Mühlethaler, and S. Boumerdassi, *Machine Learning for Networking: First International Conference, MLN 2018, Paris, France, November 27–29, 2018, Revised Selected Papers*. Paris, Frana: Springer, 2019, vol. 11407.

[27] J. Kvita, "Visualizations of rnn units," 2016.

[28] C. Olah, "Understanding lstm networks," 2015.

[29] F. S. Marumo, "Deep learning para classificação de fake news por sumarização de texto," Londrina, Paran, Brasil, 2018.

[30] L. G. Costa, "Classificação de fake news utilizando algoritmos de aprendizado de máquina e aprendizado profundo," Boa Vista, Roraima, Brasil, 2019.

[31] L. Guarise, "Deteco de notcias falsas usando tcnicas de deep learning," So Carlos, So Paulo, Brasil, 2019.

[32] F. Battisti *et al.*, "Detecção de notícias falsas utilizando aprendizado de máquina," 2020.

[33] N. Hartmann *et al.*, "Portuguese word embeddings: Evaluating on word analogies and natural language tasks," 2017.

[34] Q.-H. Vo *et al.*, "Multi-channel lstm-cnn model for vietnamese sentiment analysis." Ho Chi Minh, Vietnam: Vietnam National University, 10 2017, pp. 24–29.

[35] R. A. Monteiro *et al.*, "Contributions to the study of fake news in portuguese: New corpus and automatic detection results," in *Computational Processing of the Portuguese Language*. So Carlos - SP: Springer International Publishing, 2018, pp. 324–334.