SMS Spam Detection Through Skip-gram Embeddings and Shallow Networks

Gustavo José de Sousa, Daniel Carlos Guimarães Pedronette, João Paulo Papa, Ivan Rizzo Guilherme
São Paulo State University - UNESP, Brazil
`{gustavo.sousa,daniel.pedronette,joao.papa,ivan.guilherme}@unesp.br`

Abstract

The drastic decrease in mobile SMS costs turned phone users more prone to spam messages, usually with unwanted marketing or questionable content. As such, researchers have proposed different methods for detecting SMS spam messages. This paper presents a technique for embedding SMS messages into vector spaces that is suitable for spam detection. The proposed approach relies on mining patterns that are relevant for distinguishing spam from legitimate messages. A subset of those patterns is used to construct a function that maps text messages into a multidimensional vector space. The extracted patterns are represented as skip-grams of token attributes, where a skip-gram can be seen as a generalization of the n-gram model that allows a distance greater than one between matched tokens in the text. We evaluate the proposed approach using the generated vectors for spam classification on the UCI Spam Collection dataset. The experiments showed that our method combined with shallow networks reached accuracy that is competitive with state-of-the-art approaches.

1 Introduction

SMS spam detection is a very relevant task for mobile phone users. It can mitigate the annoyance caused by the invasive marketing applied through this platform and provide a more secure user experience by blocking potentially harmful messages. In that sense, researchers have proposed different methodologies to detect SMS spam. Abdulhamid et al. (2017) and Abayomi-Alli et al. (2019), for instance, provide comprehensive reviews of the relevant datasets and techniques found in the literature.

Most recently, some researchers have successfully applied deep learning techniques for SMS spam detection: Annareddy and Tammina (2019) provide a comparative study on using Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN); Roy et al. (2020) and Popovac et al. (2018) propose to CNN as well; Roy et al. (2020), Jain et al. (2018) and Raj et al. (2018) propose using LSTM-based architectures; Ghourabi et al. (2020) use a hybrid CNN-LSTM model; Barushka and Hajek (2018) use regularized deep multi-layer perceptrons combined with a feature selection algorithm; and Wei and Nguyen (2020) propose the use of Lightweight Gated Recurrent Units (LGRU).

Other works employ traditional classifiers to such a task (Fernandes et al., 2015; Fattahi and Mejri, 2021; Xia and Chen, 2020), including diverse models like Support Vector Machines (SVM), Hidden Markov Models (HMM), Optimum-Path Forest (OPF), k-Nearest Neighbors (KNN), decision trees, and ensembling approaches. Gupta et al. (2018) provide a comparative study using CNN and traditional machine learning architectures.

SMS spam messages have a very characteristic textual style; many inform the recipient that she has won a prize or offer apparent great deals. As such, many words (e.g., "prize", "won", "free", "bonus", etc.) can be found to be very informative for classifying a message as spam. While these words can be used to create classifiers with good performance, we observe that words and patterns of a sequence of words can be very characteristic of spam messages. For example, in the UCI SMS Spam Collection dataset (Almeida et al., 2011), the probability that a message is spam given that it contains the word "`text`" is 0.53, and the same likelihood raises to 0.96 when the message includes the pattern "`text [WORDS] to`", where "`[WORDS]`" is a placeholder for a short sequence of one or more words.

That happens because the word "`text`" can be used many times in legitimate messages, as illustrated by the following example: "*(...) I'll **text** you* *in a few (...)*". On the other hand, it is very common for spam messages to request the phone user to send a message to a certain number, like the following: "*(...)* ***text*** *TONE or FLAG* ***to*** *84199 NOW (...)*". Thus, observing this pattern gives much higher probability of a message being a spam than just finding the word "`text`" in it.

In this paper, we propose a text embedding technique for SMS messages that exploits patterns relevant for spam characterization. As a result, spam and non-spam messages are projected distinctly in the embedded space, allowing a more effective classification. The method is split into three main steps:

1. *Tokenization*: each text message from a training dataset is transformed into a sequence of sets of attributes;
2. *Skip-gram pattern mining*: the generated sequences are then processed by an algorithm for finding patterns relevant for the classification task;
3. *Embedding model*: given the set of relevant patterns, a model is constructed for mapping text messages into a vector space.

Our model presents relevant contributions, differing from some related works (especially those based on deep learning) regarding two aspects: (*i*) the proposed embedding approach is a lightweight model, being suitable to be used with shallow networks and executed on mobile devices; and (*ii*) the model allows interpretation, with explicit information associated with each dimension.

We evaluate the proposed technique using our method to generate vectors for the classification of SMS messages from the UCI SMS Spam Collection dataset (Almeida et al., 2011). Our best results surpassed the baselines from (Almeida et al., 2011) and showed to be competitive with recent deep learning approaches in terms of accuracy. In particular, our embeddings combined with a Linear SVM classifier achieved an average accuracy of 99.03% on 10-fold cross-validation experiments. Regarding efficiency aspects, prediction experiments using an ARM Cortex-A53 mobile processor yielded an average of 22 milliseconds of processing time for each SMS message.

The remainder of this paper is organized as follows: Section 2 formalizes our method and Section 3 performs an evaluation of the proposed technique. Finally, Section 4 states conclusions and presents some suggestions of future directions.

2 Formal Model Definition

This section presents a formalization of the proposed approach, which can be split into three main steps: (i) *tokenization*; (ii) *skip-gram pattern mining*; and (iii) the construction of the *embedding model*.

2.1 Tokenization

Given a set $\mathcal{X} = \{x_1, x_2, ..., x_{|\mathcal{X}|}\}$ of texts, the tokenization step constructs a set of sequences $\mathcal{S} = \{s_1, s_2, ..., s_{|\mathcal{X}|}\}$, where s_i is the sequence generated for the text x_i. We represent a given sequence as a tuple where each element is associated with a designated text fragment. Instead of merely being a substring of the text, each component of the tuple defines a set of attributes associated with the respective text fragment.

This transformation is applied with the following steps:

1. *Token splitting*: the text is split into fragments by capturing either of the following patterns, in that order of precedence: (*i*) sequence of one or more digits; (*ii*) sequence of one or more alphabetic characters; and (*iii*) sequence of any non-space character.

2. *Sub-word splitting*: each captured sequence of alphabetic characters is further broken into sub-sequences of length 3. For example, the word "`award`" is transformed into the following: "`awa`", "`war`", "`ard`". While 3 seems to be an arbitrary value for the length, we really want this parameter to be small, since that allows us to capture morphological variations of terms with similar semantics.

3. *Attribute assignment*: each fragment is transformed into a set of associated attributes, and the result of the tokenization process is the sequence of sets of attributes. Each attribute is represented as a pair (a, b), with the first element being the name of the attribute, and the second its associated value. We have three distinct attributes, i.e., $a \in \{\texttt{type}, \texttt{pos}, \texttt{sub}\}$, that can be defined as follows:

   - **`type`**: the type of the fragment. Possible values for b are `1-digit`, `2-digit`, `3-digit`, and `4+-digit` for fragments containing digits; `sub-word` for fragments that are sub-strings of alphabetic sequences; and `other` for fragments not belonging to any of the previous categories. The number before the digit types refers to the length of the captured string, `4+-digit` means 4 or more digits;

- **`pos`**: when the fragment is of type `sub-word`, b refers to the position of the sub-string in the captured word. Possible values for b are: `start`, `middle` and `end`;
- **`sub`**: when the fragment is of type `sub-word`, b is the sub-string represented by the fragment in lowercase. If the type is `other`, then b stands for the entire matched fragment.

Such a representation is coined here as an *attribute-set sequence*, and we use $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_{|\Sigma|}\}$ to denote the dictionary of attributes, that is, the set of all possible attributes.

Figure 1 illustrates the tokenization process for the string "`Call 09061701461.`" The boxes at the bottom represent the final sequence of attribute sets. For the sake of explanation, the first attribute of fragment "Cal" is the pair (**`type`**, `sub-word`).

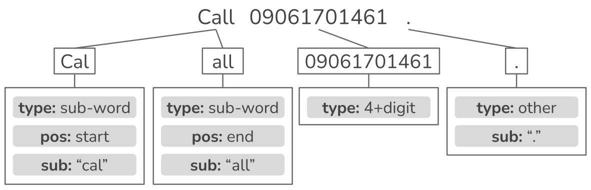

Figure 1: Illustration of the tokenization process for the string "`Call 09061701461.`".

2.2 Skip-gram pattern mining

This step assumes that each sequence from $\mathcal{S}$ is assigned to a label from a set of labels $\mathcal{Y}$. We denote $y_i = \gamma(s_i)$ as the label assigned to the sequence s_i using model γ. The objective of this step is to construct a set $\mathcal{R}$ of skip-gram patterns that are relevant for label prediction (i.e., classification).

For this work, we formally define a k-skip-n-gram pattern as a tuple $g = (g_1, g_2, ..., g_n)$ whose elements are sets of attributes. We say that g matches an attribute-set sequence $a = (a_1, a_2, ..., a_l)$ if there is a tuple $z = (z_1, z_2, ..., z_n)$ of matched indices of a, such that:

- $\forall i, 1 \leq i < n, 1 \leq z_{i+1} - z_i \leq k + 1$, that is, matched indices must be in increasing order and the distance between two matched elements must not be greater than $k + 1$;
- $\forall i, 1 \leq i \leq n, g_i \subseteq a_{z_i}$.

Note that this definition differs from the one given by Guthrie et al. (2006) in the following aspects: (i) k is the number of skips allowed between consecutive elements of the skip-gram; (ii) a match is based on set containment ($g_i \subseteq a_{z_i}$) instead of equality.

Figure 2 illustrates how the skip-gram *({(pos, start), (sub, "tex"),}, {(pos, start), (sub, "to")}, {(type, 4+-digit)})* would match the string "*text PLAY to 85222 now*". That match holds for any $k \geq 3$.

The pattern mining algorithm interactively constructs a set of skip-gram patterns that might be relevant for the classification process. At each iteration t, a set of candidate patterns $\mathcal{C}^t$ is generated and a subset $\mathcal{R}^t \subseteq \mathcal{C}^t$ is selected based on a scoring function. After T iterations, the final set $\mathcal{R}$ of patterns is then selected.

Before detailing the algorithm, we provide the following definitions, which are used throughout the remainder of this formalization:

- $\mathcal{Z}^{(g,s)}$: the set of all tuples of indices of the attribute-set sequence $s \in \mathcal{S}$ that are matched by the skip-gram pattern g. Each tuple refers to a different match. As an example, in Figure 2, $(1, 5, 6)$ would be one tuple in such a set;
- $\mathcal{M}^{(g,s)} = \{g \in \mathcal{G} : \mathcal{Z}^{(g,s)} \neq \emptyset\}$: the set of skip-gram patterns in $\mathcal{G}$ that match the sequence s;
- $P(y|g) = P(\gamma(s) = y|\mathcal{Z}^{(g,s)} \neq \emptyset)$: the estimation of the probability that the label of a sequence s is y given that the pattern g matches s, which is calculated as follows:

$$P(y|g) = \frac{|\{s \in \mathcal{S} : \gamma(s) = y \wedge \mathcal{Z}^{(g,s)} \neq \emptyset\}|}{|\{s \in \mathcal{S} : \mathcal{Z}^{(g,s)} \neq \emptyset\}|}. \quad (1)$$

- $P(g|y) = P(\mathcal{Z}^{(g,s)} \neq \emptyset|\gamma(s) = y)$: the estimation of the probability that a sequence s is matched by the pattern g given that the label of s is y. The probability is given as follows:

$$P(g|y) = \frac{|\{s \in \mathcal{S} : \gamma(s) = y \wedge \mathcal{Z}^{(g,s)} \neq \emptyset\}|}{|\{s \in \mathcal{S} : \gamma(s) = y\}|}. \quad (2)$$

- $GY(g, y) = P(y|g)^{\text{GY\_ALPHA}} P(g|y)^{1-\text{GY\_ALPHA}}$, a function to measure the dependency level between a skip-gram g and label y, we call it GY-Score. The parameter `GY_ALPHA` is a value between 0 and 1 that allows one to give more relevance to one or the other term of the multiplication;

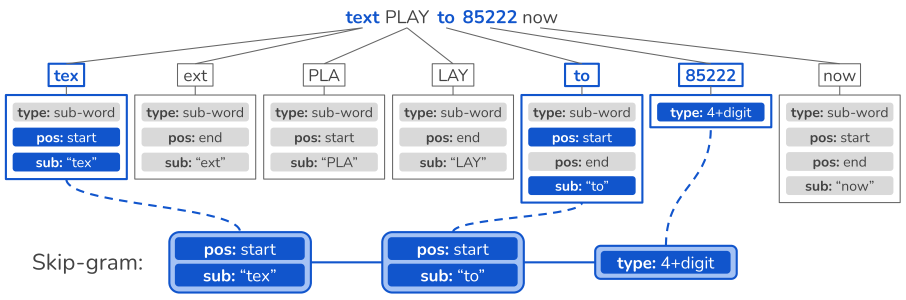

Figure 2: Example of a skip-gram matching the string "*text PLAY to 85222 now*".

- $\text{sup}(g) = |\{s \in \mathcal{S} : \mathcal{Z}^{(g,s)} \neq \emptyset\}|$, the support of skip-gram pattern g, that is, the number of elements of $\mathcal{S}$ matched by g;

We now describe the mining algorithm in the following subsections. Sections 2.2.1 and 2.2.2 detail the two steps taken at each iteration of the algorithm and Section 2.2.3 describes the final pattern selection step.

2.2.1 Candidate set construction

At each iteration t, a set $\tilde{\mathcal{C}}^t$ of new candidate skip-gram patterns is constructed. In the first iteration ($t = 0$), this set is defined as

$$\tilde{\mathcal{C}}^0 = \{(\{\sigma\}) : \sigma \in \Sigma, \text{sup}((\{\sigma\})) \geq \texttt{MIN\_SUP}\}, \quad (3)$$

that is, $\tilde{\mathcal{C}}^0$ is composed of all possible 1-skip-1-gram patterns that reach a minimum support parameter MIN_SUP.

From the second iteration and forward ($t > 0$), $\tilde{\mathcal{C}}^t$ is constructed by expanding patterns selected in the previous iteration. This is done by finding all matches between patterns selected in the previous iteration and sequences of $\mathcal{S}$, and then generating new patterns from the skip-gram respective to each match found.

For each match found, the generation of new patterns is done in two fashions: (*i*) adding new attributes found in the match either to the first or last attribute set of the pattern; and (*ii*) either appending or prepending singleton sets with attributes found around the match and, as such, increasing the size of the new pattern by 1.

All the newly generated patterns are filtered so that only those that reach the minimum support parameter MIN_SUP and that improve the GY-Score by at least MIN_GY_DIFF are kept in $\tilde{\mathcal{C}}^t$.

Formally, for $t > 0$, we can define:

$$\tilde{\mathcal{C}}^t = \bigcup_{s \in \mathcal{S}} \mathcal{E}^{(\mathcal{R}^{t-1}, s)}, \quad (4)$$

where $\mathcal{R}^{t-1}$ represents the set of patterns selected in the previous iteration and $\mathcal{E}^{(\mathcal{G},s)}$ can be described as the set of patterns generated as expansions of patterns in $\mathcal{G}$ for matches found in the sequence s, which can be defined as:

$$\begin{aligned}\mathcal{E}^{(\mathcal{G},s)} = \{g' : (g', g) \in \bigcup_{a,b \in \{1,2\}} \mathcal{E}_{ab}^{(G,s)},\\ GY(g', \gamma(s)) - GY(g, \gamma(s)) \geq \texttt{MGY},\\ \text{sup}(g') \geq \texttt{MIN\_SUP}\}.\end{aligned} \quad (5)$$

The sets given by each $\mathcal{E}_{ab}$ represent different types of expansions. Each element of such sets is a pair in the form (g', g), with g' being the newly generated skip-gram pattern and g, the original (from the match). The term MGY is an abbreviation for the MIN_GY_DIFF parameter.

Specifically, the sets represented by $\mathcal{E}_{11}^{(\mathcal{G},s)}$ and $\mathcal{E}_{12}^{(\mathcal{G},s)}$ are the expansions generated using the first method, by adding attributes to the attribute-sets on the extremes on the patterns:

$$\begin{aligned}\mathcal{E}_{11}^{(\mathcal{G},s)} = \{(g', g) : g' = (g_1 \cup \{\sigma\}, ..., g_n),\\ g = (g_1, ..., g_n) \in \mathcal{M}^{(G,s)},\\ s = (a_1, ..., a_m),\\ (j, ...) \in \mathcal{Z}^{(g,s)},\\ \sigma \in a_j - g_1\}\end{aligned} \quad (6)$$

$$\begin{aligned}\mathcal{E}_{12}^{(\mathcal{G},s)} = \{(g', g) : g' = (g_1, ..., g_n \cup \{\sigma\}),\\ g = (g_1, ..., g_n) \in \mathcal{M}^{(G,s)},\\ s = (a_1, ..., a_m),\\ (..., j) \in \mathcal{Z}^{(g,s)},\\ \sigma \in a_j - g_n\}\end{aligned} \quad (7)$$

On the other hand, $\mathcal{E}_{21}^{(\mathcal{G},s)}$ and $\mathcal{E}_{22}^{(\mathcal{G},s)}$ apply the second method of expansion, by prepending and appending elements to the skip-grams:

$$\begin{aligned}\mathcal{E}_{21}^{(G,s)} = \{(g', g) : g' = (\{\sigma\}, g_1, ..., g_n),\\ g = (g_1, ..., g_n) \in \mathcal{M}^{(G,s)},\\ s = (a_1, ..., a_m),\\ (j, ...) \in \mathcal{Z}^{(g,s)},\\ \max\{1, j - k - 1\} \leq l < j,\\ \sigma \in a_l\}\end{aligned} \quad (8)$$

$$\begin{aligned}\mathcal{E}_{22}^{(G,s)} = \{(g', g) : \; & g' = (g_1, ..., g_n, \{\sigma\}),\\ & g = (g_1, ..., g_n) \in \mathcal{M}^{(G,s)},\\ & s = (a_1, ..., a_m),\\ & (..., j) \in \mathcal{Z}^{(g,s)},\\ & j < l \leq \min\{m, j+k+1\},\\ & \sigma \in a_l\}\end{aligned} \tag{9}$$

, where k refers to the maximum number of skips for our the skip-gram model, which is referred by the parameter SKIPGRAM_K in the evaluation section.

The final set of candidate patterns $\mathcal{C}^t$ for the iteration t is then composed of the candidates from this and previous iterations that have not already been selected:

$$\mathcal{C}^t = \bigcup_{j=t-\text{CBS}+1}^{t} \tilde{\mathcal{C}}^j - \bigcup_{j=0}^{t-1} \mathcal{R}^j, \tag{10}$$

where CBS is an abbreviation for CAND_BUFFER_SIZE, which is a parameter that defines how many of the new candidate sets from previous iterations are remembered at the iteration t. This parameter gives candidates from the $\text{CBS} - 1$ previous iterations not yet selected a chance of being selected before they are "forgotten".

2.2.2 Pattern selection

Given the set of candidates $\mathcal{C}^t$, a set of new selected patterns $\mathcal{R}^t \subseteq \mathcal{C}^t$ is constructed by iterating over the sequences in $\mathcal{S}$ and selecting matching patterns with the best GY-Scores. Formally, the set $\mathcal{R}^t$ can be defined by

$$\mathcal{R}^t = \bigcup_{s \in \mathcal{S}} \text{Select}(\mathcal{C}^t, s), \tag{11}$$

where

$$\text{Select}(\mathcal{C}^t, s) = \operatorname*{arg\,max}_{g \in \mathcal{M}^{(\mathcal{C}^t, s)}} GY(g, \gamma(s)) \tag{12}$$

is either a set containing one of the patterns in $\mathcal{C}^t$ matching s with the highest GY-Score or the empty set if no matching pattern is found.

2.2.3 Final selection

After T iterations, a last step is performed in order to filter the selected patterns to contain only those that hit a minimum threshold value of information gain, given by a parameter MIN_IG, and then that filtered set is used to selected the best FINAL_SELECT_K matched patterns for each sequence. Thus, the final set $\mathcal{R}$ of selected skip-gram patterns is defined by:

$$\mathcal{R} = \bigcup_{s \in \mathcal{S}} \text{Best}(\mathcal{R}_F, s), \tag{13}$$

$$\mathcal{R}_F = \left\{ g \in \bigcup_{j=0}^{T-1} \mathcal{R}^j : IG(g) \leq \text{MIN\_IG} \right\}, \tag{14}$$

$$IG(g) = \sum_{y \in \mathcal{Y}} P(y|g) \cdot \ln\left(\frac{P(y|g)}{P(y)}\right), \tag{15}$$

where $\text{Best}(\mathcal{R}_F, s)$ is a set containing the top FINAL_SELECT_K elements from $\mathcal{M}^{(\mathcal{R}_F, s)}$ with the greatest GY-Scores; and $P(y)$ is the marginal probability that a sequence has label y and is estimated with:

$$P(y) = \frac{|\{s \in \mathcal{S} : \gamma(s) = y\}|}{|\mathcal{S}|}. \tag{16}$$

The number of iterations taken by the algorithm (i.e. T) is referred by the parameter ITERATIONS in the evaluation section.

2.3 Embedding Model

Once a set $\mathcal{R}$ of relevant skip-grams is found, it can then be used to build an embedding model. Our driving hypothesis is:

1. that the presence or absence of one or more of those patterns in a sequence greatly changes prior probability of labels;

2. that each pattern carries a very characteristic contextual baggage, in the sense that there is some level of statistical dependency between the occurrence of a pattern and of other attributes in a sequence.

As such, we propose an embedding function $v(s)$ that exploits both the information about the presence of patterns in the sequence s and the contextual information associated with each pattern taken into consideration.

We formalize our embedding model as follows. First we define a subset $\mathcal{R}_d$ of d skip-gram patterns from $\mathcal{R}$ to be used for the embedding. Then we define the function $v(s)$ in terms of elements of $\mathcal{R}_d$. The value of d is referred by the parameter DIMENSIONS in the evaluation section.

Let $\mathcal{R}_d \subseteq \mathcal{R}$ be a set containing the d most informative skip-gram patterns, that is: (*i*) $|\mathcal{R}_d| = d$, and (*ii*) $\forall r \in \mathcal{R}_d, \forall r' \in \mathcal{R} - \mathcal{R}_d, IG(r) \geq$

$IG(r')$. Each element of $\mathcal{R}_d$ will be associated with one dimension of the target vector space. For that, we select an arbitrary ordering of elements of $\mathcal{R}_d$, which we denote by the following indexing:

$$\mathcal{R}_d = \{r_1, r_2, ..., r_d\}. \tag{17}$$

Now, we define the value of $v(s)$ to be a d-dimensional vector as follows:

$$v(s) = \alpha \cdot u_1(s) + (1 - \alpha) \cdot u_2(s) \tag{18}$$

$$u_1(s) = [u_{11}(s), u_{12}(s), ..., u_{1d}(s)] \tag{19}$$

$$u_2(s) = [u_{21}(s), u_{22}(s), ..., u_{2d}(s)], \tag{20}$$

where:

- $u_1(s)$ is the vector accounting for skip-gram matches found in s. Each component $u_{1j}(s)$ is defined as:

$$u_{1j}(s) = \begin{cases} 1 & \text{if } r_j \in \mathcal{M}^{(\mathcal{R}_d, s)} \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

- $u_2(s)$ is the vector accounting for contextual information. Each component $u_{2j}(s)$ must be a measure that, in a way, reflects the statistical dependence between the pattern r_j and attributes found in s. For this work, we choose a very simple approach by taking the average of probabilities of r_j having a match with s given each relevant attribute of s. Thus, we define:

$$u_{2j}(s) = \frac{1}{|\mathcal{A}^s|} \sum_{\sigma \in \mathcal{A}^s} P(r_j | (\{\sigma\})), \tag{22}$$

  where $\mathcal{A}^s$ is the set of every attribute σ found in s in such that $\sup((\{\sigma\})) \leq$ MIN_U2_SUP, and MIN_U2_SUP is a parameter for filtering out rare attributes;

- $\alpha \in [0, 1]$ is a parameter for weighting the two types of information being aggregated in $v(s)$. We refer to this parameter by U_ALPHA in the evaluation section.

3 Evaluation

This section describes the evaluation of our proposed technique. We used the UCI SMS Spam Collection[1] (Almeida et al., 2011), which is a dataset comprised of 4,827 (86.6%) legitimate messages and 747 (13.4%) spam messages, with an average length of 16±12 tokens per message.

Parameter	Values used
Skip-gram mining parameters	
SKIPGRAM_K	6
GY_ALPHA	0.8
MIN_SUP	10
MIN_GY_DIFF	0.05
CAND_BUFFER_SIZE	6
ITERATIONS	6
MIN_IG	0.5
FINAL_SELECT_K	5
Embedding parameters	
U_ALPHA	0.0 0.1 ... 0.9 1.0
MIN_U2_SUPPORT	10 20 30
DIMENSIONS	50 100 189[2]

Table 1: Variations of parameters used for generating the embedding models.

We first conducted our evaluation by following the same experimental protocol used by the authors for their classification baselines and provide the analysis of our results in this in Sections 3.1 and 3.2. We used the train split to construct different variations of our embedding model and then evaluated three different machine learning methods for classification of spam and non-spam message. Table 1 displays the variation of parameters used for our experiments. The parameters related to the skip-gram mining step were fixed at single values after some empirical preliminary experimentation.

Later, we extended our evaluation by using different split configurations of the dataset in order to provide comparisons of our model with related work, which are presented in Section 3.2.1. Finally, Section 3.3 presents a visual evaluation of generated embeddings.

3.1 Selected patterns

The skip-gram pattern mining step applied to the training dataset resulted in 189 selected patterns. Table 2 displays the distribution of the types of patterns selected as well as the average information gain scores. We can note that longer patterns tend to be more informative but rarer at the same time. Although patterns with $n > 3$ were allowed (because ITERATIONS $=$ 6 and CBS $=$ 6), only patterns of length 1, 2 and 3 were selected.

Some interesting examples from the patterns with the highest information gain scores are listed

[1]Available at https://archive.ics.uci.edu/ml/datasets/sms+spam+collection.

[2]We use 189 instead of 200 here because that was the total number of selected skip-gram patterns after the mining process.

Type	Proportion (%)	Avg. Info. Gain
6-skip-1-gram	44.44	1.07±0.41
6-skip-2-gram	51.86	1.42±0.39
6-skip-3-gram	3.70	1.77±0.23

Table 2: Distribution of types of skip-gram patterns selected.

in Table 3. The third column displays the estimated conditional probability that a message is a spam given that the pattern matches it, and the fourth column shows the proportion of the spam messages that are covered by the respective pattern. The pattern `[{(sub, "cal")}, {(type, 4+-digit)}]` showed to be a very recurrent one, representing about 32% of the spam messages in the training dataset.

3.2 Classification

We used three different types of classifiers to assess the effectiveness of our embedding model: Nearest Neighbors Classifier, Multilayer Perceptron (MLP) and Linear Support Vector Machine (LSVM). For the three methods, we used the implementation provided by the Scikit-learn library (Pedregosa et al., 2011)[3].

Table 4 summarizes the best result achieved for each classification method. The best baseline found by (Almeida et al., 2011) is also shown in the table for comparison purposes. The table has the following columns:

- *Method*: name of the method used for classification;
- *SC*: percentage of spam caught;
- *BH*: percentage of blocked hams, that is, legitimate messages that were classified as spam;
- *Acc*: accuracy of the classification in percentage unit;
- *MCC*: the Matthews Correlation Coefficient (Matthews, 1975), which is a good metric to be used for unbalanced datasets (as it is the case with the dataset used in this work). The table is sorted by this column in descending order.

The MLP model was constructed using two layers with 100 neurons in each layer and the logistic function for activation, and it was optimized using the Adam optimizer (Kingma and Ba, 2014). Our preliminary experiments showed that using two layers yielded better results than using just one; and increasing further the number of layers hurt the performance of the model. Each embedding parameter variation was executed 10 times and the average of the metrics were taken.

For the KNN model, we varied the number of neighbors used by 5, 10 and 15 and tested both uniform and distance-based weighting schemes. We found that using 5 neighbors and distance-based weighting yielded the best result for this type of classifier.

We used the default parameters provided by Scikit-learn for the LSVM model. It is worth mentioning that the reported baseline also uses linear support vector machines.

When compared with the baseline, the results in Table 4 show that our embedding technique combined with those classification methods displayed significant increases in the rate of detected spams while keeping very low increases in the rate of blocked hams (less than 0.5%).

Table 5 shows the embedding parameters respective to the results in Table 4. The values are very similar between different methods. It is a consensus for all evaluated classification techniques that low values for `U_ALPHA` tend to yield the best results for classification tasks, which suggests that giving more weight to contextual information associated with skip-gram patterns is more beneficial.

3.2.1 Comparison with related work

We also compared our approach with recent deep learning techniques found in the recent literature. Since they used different experimental protocols, we re-executed our experiments using different splits of the dataset in order to match the different protocols used by related works. Table 6 displays the performance of our best models and related works grouped by the experimental protocol and Table 7 shows the best embedding parameters found for each protocol. The comparison displayed in Table 6 shows that our embedding technique, combined with a Linear SVM, is competitive with the deep learning approaches proposed recently.

3.3 Visualization

We finalize our evaluation by providing a visual analysis of the embeddings generated with the parameters from Table 5. For that, we used UMAP (McInnes et al., 2018a,b) to map each embedding generated for the test split into a 2-dimensional space and plotted the results. Figure 3

[3]We used version 0.24.0 of the library

g	$IG(g)$	$P(y = \text{spam} \mid g)$	$P(g \mid y = \text{spam})$
`[{(sub, "cal")}, {(type, 4+-digit)}]`	1.951	1.0	0.319
`[{(sub, "cla")}, {(sub, "aim")}]`	1.951	1.0	0.156
`[{(sub, "pri")}, {(sub, "ize")}]`	1.951	1.0	0.109
`[{(sub, "tex")}, {(sub, "to")}, {(type, 4+-digit)}]`	1.951	1.0	0.088

Table 3: Examples of selected patterns.

Method	SC(%)	BH(%)	Acc(%)	MCC
MLP	$90.12_{\pm 0.41}$	$0.61_{\pm 0.05}$	$98.19_{\pm 0.06}$	$0.919_{\pm 0.003}$
KNN	88.41	0.38	98.15	0.917
LSVM	87.62	0.29	98.13	0.916
Baseline	83.10	0.18	97.64	0.893

Table 4: Summary of results for classification.

Method	U_ALPHA	MIN_U2_SUPPORT	DIMENSIONS
MLP	0.3	30	189
KNN	0.1	30	189
LSVM	0.2	10	189

Table 5: Best embedding parameters for classification.

provides the visualization for the embeddings generated for the MLP classifier from Table 4. We can see that the projection successfully separates most of the vectors representing spams from the non-spam ones. Interestingly, some small clusters can be observed for both classes, which could indicate that, inside each class, further categorization could be possible.

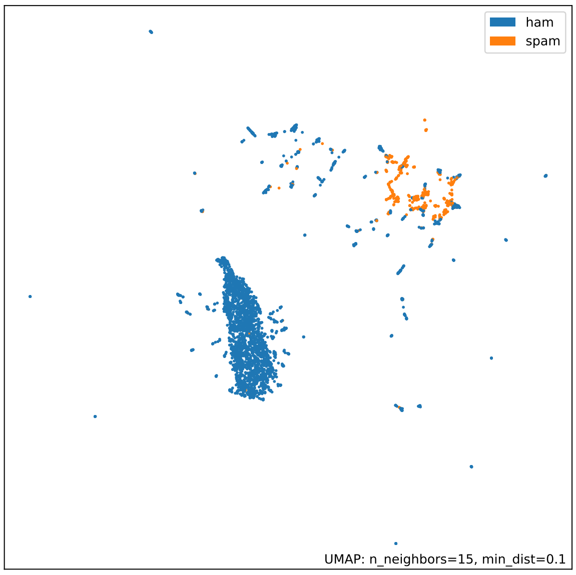

Figure 3: Visualization of embeddings for the MLP classifier.

4 Conclusions and Future Works

In this work, we proposed an embedding technique for short texts based on skip-gram patterns and successfully evaluated our method in a dataset for SMS spam detection. Our experiments showed that our method, when combined with "shallow" classifiers, is competitive with recent deep learning approaches. Our results suggest that using this kind of pattern matching might be a good way to improve representations for textual classification tasks.

We see some interesting ways of improving the work presented in this paper, which can be taken as future work, such as:

- increasing our evaluation scope by adding more textual datasets (not limited to SMS) to our evaluation workload;
- investigating ways to integrate word vectors in our skip-gram matching mechanism in order to be able to capture patterns that do not necessarily have the same attributes but that are semantically similar;

Reference	Classifier	Accuracy (%)
	10-fold cross-validation	
Roy et al. (2020)	CNN	99.44
This work	LSVM	$99.03_{\pm 0.31}$
Barushka and Hajek (2018)	DBB-RDNN-Rel	98.51
Popovac et al. (2018)	CNN	98.4
	Holdout 4:1	
This work	LSVM	99.1
Gupta et al. (2018)	CNN	99.1
Wei and Nguyen (2020)	LGRU	99.04
Annareddy and Tammina (2019)	RNN	97.8
	Holdout 3:1	
This work	LSVM	98.99
Raj et al. (2018)	LSTM	97.5
	Holdout 2:1	
This work	LSVM	98.76
Roy et al. (2020)	CNN	98.63

Table 6: Comparison with related work.

Protocol	U_ALPHA	MIN_U2_SUPPORT	DIMENSIONS
10-fold	0.3	10	300
Holdout 4:1	0.2	10	300
Holdout 3:1	0.2	10	300
Holdout 2:1	0.2	10	300

Table 7: Embedding parameters concerning the results found in Table 6.

- looking for alternative ways of defining the contextual component of the embedding function, namely, the function $u_2(s)$. In this work we chose to use a simple model that takes the average of conditional probabilities - more robust predictive models could be used instead.

Acknowledgments

The authors are grateful to the São Paulo Research Foundation - FAPESP (grants #2014/12236-1, #2018/15597-6, and #2019/07665-4), Brazilian National Council for Scientific and Technological Development - CNPq (grants #307066/2017-7 and #309439/2020-5), and Petrobras (grant #2019/00697-8).

References

Olusola Abayomi-Alli, Sanjay Misra, Adebayo Abayomi-Alli, and Modupe Odusami. 2019. A review of soft techniques for sms spam classification: Methods, approaches and applications. *Engineering Applications of Artificial Intelligence*, 86:197–212.

S. M. Abdulhamid, M. S. Abd Latiff, H. Chiroma, O. Osho, G. Abdul-Salaam, A. I. Abubakar, and T. Herawan. 2017. A review on mobile sms spam filtering techniques. *IEEE Access*, 5:15650–15666.

Tiago A. Almeida, José María G. Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of sms spam filtering: New collection and results. In *Proceedings of the 11th ACM Symposium on Document Engineering*, DocEng '11, page 259–262, New York, NY, USA. Association for Computing Machinery.

Sunil Annareddy and Srikanth Tammina. 2019. A comparative study of deep learning methods for spam detection. In *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. IEEE.

Aliaksandr Barushka and Petr Hajek. 2018. Spam filtering using integrated distribution-based balancing approach and regularized deep neural networks. *Applied Intelligence*, 48(10):3538–3556.

Jaouhar Fattahi and Mohamed Mejri. 2021. Spaml: a bimodal ensemble learning spam detector based on nlp techniques. In *5th International Conference on Cryptography, Security and Privacy*. IEEE.

D. Fernandes, K. A. P. Da Costa, T. A. Almeida, and J. P. Papa. 2015. Sms spam filtering through optimum-path forest-based classifiers. In *IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 133–137.

Abdallah Ghourabi, Mahmood A. Mahmood, and Qusay M. Alzubi. 2020. A hybrid cnn-lstm model for sms spam detection in arabic and english messages. *Future Internet*, 12(9):156.

Mehul Gupta, Aditya Bakliwal, Shubhangi Agarwal, and Pulkit Mehndiratta. 2018. A comparative study of spam sms detection using machine learning classifiers. In *2018 Eleventh International Conference on Contemporary Computing (IC3)*. IEEE.

David Guthrie, Ben Allison, Wei Liu, Louise Guthrie, and Yorick Wilks. 2006. A closer look at skip-gram modelling. In *LREC*, volume 6, pages 1222–1225.

Gauri Jain, Manisha Sharma, and Basant Agarwal. 2018. Optimizing semantic lstm for spam detection. *International Journal of Information Technology*, 11(2):239–250.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

B.W. Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, 405(2):442 – 451.

Leland McInnes, John Healy, and James Melville. 2018a. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. 2018b. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Milivoje Popovac, Mirjana Karanovic, Srdjan Sladojevic, Marko Arsenovic, and Andras Anderla. 2018. Convolutional neural network based sms spam detection. In *2018 26th Telecommunications Forum (TELFOR)*. IEEE.

Hans Raj, Yao Weihong, Santosh Kumar Banbhrani, and Soomro Pir Dino. 2018. Lstm based short message service (sms) modeling for spam classification. In *International Conference on Machine Learning Technologies - ICMLT 18*. ACM Press.

Pradeep Kumar Roy, Jyoti Prakash Singh, and Snehasish Banerjee. 2020. Deep learning to filter sms spam. *Future Generation Computer Systems*, 102:524–533.

Feng Wei and Trang Nguyen. 2020. A lightweight deep neural model for sms spam detection. In *2020 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE.

Tian Xia and Xuemin Chen. 2020. A discrete hidden markov model for sms spam detection. *Applied Sciences*, 10(14):5011.