



**Text Analysis and Retrieval 2020**  
**Course Project Reports**

University of Zagreb  
Faculty of Electrical Engineering and Computing

This work is licensed under the Creative Commons Attribution – ShareAlike 4.0 International.

<https://creativecommons.org/licenses/by-sa/4.0/>



**Publisher:**

University of Zagreb, Faculty of Electrical Engineering and Computing

**Organizers & editors:**

Mladen Karan  
Jan Šnajder

**Reviewers:**

Alen Carin  
Mladen Karan  
Doria Šarić  
Jan Šnajder  
Lucija Šošić  
Donik Vršnak  
Davor Vukadin  
Zorica Žitko

**ISBN 978-953-184-270-9**

**Course website:**

<http://www.fer.unizg.hr/predmet/apt>

**Powered by:**

Text Analysis and Knowledge Engineering Lab  
University of Zagreb  
Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia  
<http://takelab.fer.hr>



**TakeLab**

# Preface

This is the seventh booklet in a series of students' project reports from the Text Analysis and Retrieval (TAR) course, taught at the Faculty of Electrical Engineering and Computing (FER), University of Zagreb. TAR teaches the foundations of natural language processing and information retrieval, with a focus on practical applications, all while being in line with best practices in the area.

These outcomes are achieved through hands-on student projects, which are the central part of the course. Students that complete this course gain both practical and theoretical skills in data science and machine learning, with a strong focus on text data. Given the bewildering and ever-growing amount of information available today (in text form especially), such skills are invaluable to employers. We are happy and proud to supply our students with such an extra edge on the increasingly competitive job market.

This booklet represents the results of 15 projects (with one additional project left out as it got accepted and published at another venue), which are the work of 48 students. Most of the topics were adopted from recent workshops and shared tasks like SemEval and CLEF. However, some students proposed their own, which resulted in a very diverse set of topics, including propaganda detection, bot detection, cybersecurity, emoji analysis, and depression detection, to name a few. In view of the Covid-19 pandemic that occurred in 2020. There were also two teams that investigated ways to use NLP techniques to aid analyzing and searching research literature pertaining to the virus. With respect to methods, the trend of increasing popularity for deep learning is present for several years and continues this year, with most teams using some sort of deep learning. We believe part of the reason for this is the increasing availability of deep learning libraries that are simple to use and the recent development of contextualized word embedding models that provide a high quality pretrained neural word representations useful for almost any task. Some of the teams really went above and beyond by implementing custom neural architectures with custom loss functions etc. As in the previous years, the project reports were written in the form of a research paper. The purpose of this was to both teach the students to effectively present their results, as well as to give them a feeling of how actual scientific research works. To this end, in addition to writing a project report, the students also actively participated in several paper reading sessions, where scientific papers were discussed in groups. As students seldom get an opportunity to get involved in hands-on scientific research during their Master's programme, we felt this would be a valuable new experience for them. Similarly to previous editions of the course, our intuitions about this approach were confirmed by resoundingly positive student feedback.

This year's edition of TAR was held completely on-line, due to the COVID-19 pandemic. While this required some organizational changes, we believe we have managed to preserve the intended experience as much as possible under the circumstances. This was facilitated by considerable patience and understanding on part of the students. As the course organizers, we thank the students for demonstrating remarkable motivation and enthusiasm throughout the course. We are honored to have had the opportunity to work with them.

*Mladen Karan and Jan Šnajder*

# Contents

<i>Sentence-Level Detection of Propaganda in News Articles</i>	
Ivan Almer, Sven Goluža, Tina Mustapić . . . . .	1
<i>Bot Detection From a Single Tweet</i>	
Ivan Anić, Ivan Bilić, Silvije Škudar . . . . .	5
<i>To Context or Not to Context? Analysis of Non-contextualized Word Embeddings in Propaganda Detection Task</i>	
Luka Barišić, Marko Pisačić, Filip Radović . . . . .	9
<i>Using Additional Features in Emoji Prediction</i>	
Bruno Belić, Simon Grgurina, Barbara Lugar . . . . .	13
<i>Why Is Emoji Prediction Difficult?</i>	
Matija Bertović, Antun Magdić, Ante Žužul . . . . .	17
<i>Depression Detection in Online Forums using Stylistic Features and Random Sampling</i>	
Vedran Bogdanović, Adi Caušević, Juraj Vladika . . . . .	22
<i>COVID-19 Information, Where Are You Hiding?</i>	
Marko Cvitković, Nikolina Čović, Patrik Marić . . . . .	27
<i>How Offensive is Negative Sentiment? Using Offensive Language Detection Models for Sentiment Classification</i>	
Katarina Čavar, Sanja Deur, Dorotea Protrka . . . . .	32
<i>How shallow are bots?</i>	
Gabriela Dorić, Ivica Duspara, Marko Kršić . . . . .	37
<i>SMS Spam Detection</i>	
Barthélémy Marsault, Florian Gigot, Gaéтан Jagorel . . . . .	42
<i>Semantic Extraction from Cybersecurity Reports (SECR)</i>	
Markus Paulsen, Luis Fernandez, Bingji Wang . . . . .	46
<i>Don't worry, I am not depressed... Or am I?</i>	
Luka Pavlović, Martin Sršen, Krunoslav Jurčić . . . . .	51
<i>Propaganda in Press: Challenges of Automatic Detection</i>	
Dora Pušelj, Tena Škalec . . . . .	55
<i>Search Less, Research More: Improving Information Retrieval in the Face of COVID-19</i>	
Mario Šaško, Ivan Lovrenčić, Nikola Buhiniček . . . . .	60
<i>Detection of Propaganda Techniques in News Articles</i>	
Karol Wilk . . . . .	64

# Sentence-Level Detection of Propaganda in News Articles

Ivan Almer, Sven Goluža, Tina Mustapić

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia  
{ivan.almer, sven.goluza, tina.mustapic}@fer.hr

## Abstract

Propaganda detection in news and media is a tough task even for humans. Nevertheless, intelligent systems could be beneficial in detecting propagandistic content in news articles. Due to more attention given to article-level propaganda detection, we decided to go for sentence-level detection and provide readers with an opportunity to see what makes the article propagandistic. Our goal was to demonstrate the difference in the performance of two seemingly similar models, a stacked model versus a standalone model, and show why one has proven itself to be better for the task of detection of propagandistic content in a sentence. To support our claims, we ran two experiments on a corpus of 371 news articles.

## 1. Introduction

Webster’s dictionary defines propaganda as the spreading of ideas, information, or rumors for the purpose of helping or injuring an institution, a cause, or a person. The word propaganda is often negatively linked to politics and comes in various media such as the press. Most work involving propaganda detection is article-level (Barrón-Cedeno et al., 2019), meaning that the only information we get is if the article is propaganda or not, but we don’t know what content in the article is misinformation. For this reason it is interesting for us to know the exact propagandistic content in an article. Sometimes it’s hard to differentiate between the author’s subjectiveness on the topic and actual idea pushing. Another problem we have to tackle is annotator subjectiveness (Martino et al., 2019). The average annotator cannot detach his mindset from the judgment of propaganda and bias, for example, if a propagandistic article expresses ideas aligned with the annotator’s beliefs, it is unlikely that he would judge it as such.

Taking into account all problems of data annotation and that almost all datasets for this problem are imbalanced, we wanted to investigate performance of different systems on propaganda detection task. Generally, sentence-level propaganda detection systems are made of multiple components where error propagation between components can negatively influence model’s performance. We argue that error propagation comes from losing important training information between components of the multi-component system. We built two very similar recurrent neural network (RNN) models with the difference being that one model was a multi-component system and the other was standalone. We conducted two experiments that involved sentence-level propaganda detection and demonstrated that the standalone model outperformed the multi-component one on sequence labelling task.

## 2. Related Work

The task was initially introduced in (Da San Martino et al., 2019a) for the NLP4IF 2019 competition. The goal of the shared task was to produce models capable of spotting sentences and text fragments in which propaganda tech-

niques are used in a news article. There were two sub-tasks, one of which was a sentence-level binary classification task (SLC) asking to detect the sentences that contain propaganda. Most teams based their approach on fine-tuned BERT for the SLC task in combination with LSTMs, hand-crafted features and embeddings, with the winning team using BERT trained on Wikipedia and BookCorpus. The follow-up competition was SemEval 2020<sup>1</sup> which is based on the same corpus, but expanded. Its shared task is propagandistic span detection and classification but unfortunately, the results have not been published yet. We would like to point out that our sequence labeling results and SLC results are not directly comparable to the NLP4IF 2019 competition because our datasets differ, also the comparison to competition results was not our goal. Our standalone model did achieve good results in the modified task 1 of SemEval 2020 but they also aren’t comparable (see Section 6.).

## 3. Dataset

The input for both tasks consists of news articles from the Propaganda Techniques Corpus (PTC), used at SemEval 2020 task 11 (Da San Martino et al., 2019b). PTC is a corpus of news articles manually annotated by six professional annotators. The articles were retrieved with the newspaper3k library. The title is in the first row, followed by an empty row, and the article content starts from the third row, one sentence per row. PTC includes one plain text file and two annotation files per document. One annotation file consists of the article id, the beginning of the propagandistic span and the end of the propagandistic span for every propagandistic span in the article. Some spans may overlap for which we proposed a solution in Section 4.1. The second annotation file is identical to the first, except that it includes the type of propaganda used in every propagandistic span. For our models, we use the first annotation file. Partitions for training and development of PTC include 446 articles (350k tokens), of which 371 were labelled, from 48 news outlets. Because the dataset is quite imbalanced (see Table 1) we decided to split every article into sentences and

---

<sup>1</sup>SemEval 2020 task 11

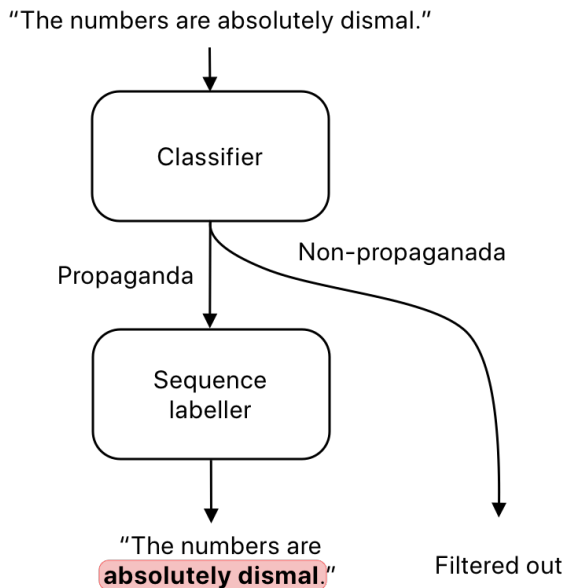


Figure 1: Stacked system

adjust the annotated propagandistic spans from where the propaganda extends in the article to where it extends in the sentence. This reduces the class imbalance but the results are still not optimal.

#### 4. Experimental Setup

While examining the dataset it was obvious that propagandistic sentences are rare, and what is more, propagandistic content takes up approximately 12% of total corpus. The mentioned imbalance between propagandistic and non-propagandistic content could be a great obstacle for sequence labeling models. This nudged us to come up with a possible solution to the problem. We tackle the task of propaganda detection by setting up and comparing two separate systems.

The first system is a two-component system - a stacked system (Figure 1). The first component is a binary classifier which labels each sentence as propagandistic or non-propagandistic based on the appearance of propagandistic phrases in it. The second component deals with the sequence labeling problem where each token is labeled with true if it is included in the propaganda span or false if it's not. By filtering out sentences in the described dataset and feeding propagandistic sentences as an input to the sequence labeling task, we want to see if such a system outperforms the standalone model (Figure 2). The mentioned standalone system deals with sequence labeling span identification and is trained on the original dataset that contains all sentences, including non-propagandistic ones.

##### 4.1. Preprocessing

The dataset was initially very well organized but some articles contained overlapping spans. This could give the model a hard time since it will have to label some spans with multiple labels. We decided to merge those overlapping spans in a single label called **Multiple-Labels**, which indicates that the specified span contains two or more types

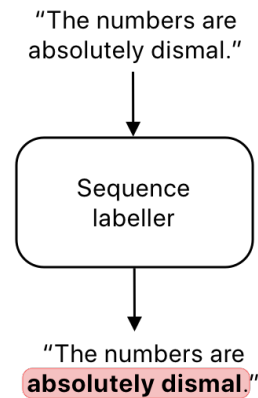


Figure 2: Standalone model

of propaganda. By doing that, an interesting thing happened and some labels were completely lost meaning some labels came solely as a subpart of other labels.

##### 4.2. Proposed Model Architectures

We opted to go “deep” as deep learning models give better performance than traditional shallow models for sequence labeling, such as the Maximum Entropy Markov Model and Conditional Random Fields. The stacked system and standalone model have a similar architecture which is based on LSTM cells (Hochreiter and Schmidhuber, 1997) that deal well with the vanishing gradient problem and retain information for a long period.

**The stacked system** consists of two specific types of recurrent neural networks. The first component is a sentence-level classifier and its input consists of pre-trained 300-dimensional word embeddings from the tokenized sentence. They are fed to a Bidirectional LSTM layer that consists of a 64-dimensional vector of hidden units. Output vectors from forward and backward layers are concatenated to a 128-dimensional vector. A dropout layer with a rate of 0.5 is applied to BiLSTM output to avoid overfitting. Output of this layer is fed to a sigmoid classifier that labels sentences as *propagandistic* or *non-propagandistic*. If the sentence is labeled as non-propagandistic, each token in it is automatically labeled as non-propagandistic, otherwise, tokenized sentence is passed to the second component. The second component is a many-to-many RNN architecture, which again combines word embeddings as an input to the BiLSTM model for a sequence labeling task. The BiLSTM layer is identical as the first one, except it has concatenated output vectors for each input which are fed to a softmax classifier that labels each token as propagandistic, non-propagandistic or as padding. We experimented with adding CRF layer on the output of BiLSTM layer, but BiLSTM-CRF architecture didn't lead to better results compared with BiLSTM-softmax local classification.

**The standalone model** has the same architecture as the second component of the stacked system, with a few fine-tuned hyperparameters. Its BiLSTM layer has a hidden unit size of 128 dimensions. Output vectors from two LSTM layers are summed to form the output of the BiLSTM layer which is then fed to a softmax classifier. Both models are

Table 1: Dataset statistics

Class	Article	Sentence
Propaganda	357 (96.2%)	4643 (28.5%)
Non propaganda	14 (3.8%)	11676 (71.5%)

implemented with Keras API<sup>2</sup> that is running on top of the machine learning library TensorFlow.

### 4.3. Training

Before a sentence is given to the model it has to be tokenized. When the first component of our stacked system, sentence classifier, uses a sentence as an input, we tokenize its text using spaCy’s<sup>3</sup> tokenizer, remove stop words and lemmatize each token.

Both sequence labelling models, the one in the stacked system and the standalone one, get tokenized sentences where each token is lemmatized, but no words are removed since each token has its label.

For word embeddings, we use publicly available GloVe word embeddings (Pennington et al., 2014) trained on 6 billion words as our word vector representation. After trying out different vector dimensions, we opted to choose 300-dimensional word vectors as our model input embeddings.

Our optimization algorithm was Adam (Kingma and Ba, 2014) with default Keras hyperparameters, which minimizes binary cross-entropy loss (for sigmoid classifier) and categorical cross-entropy loss (for softmax classifier).

### 4.4. Experimental Tasks

To prove our argument that implementing a stacked system for propaganda detection will decrease model performance we conducted two experiments. The first experiment deals with **span identification** or, to be more precise, with identifying propagandistic fragments in sentences. Given a sentence, the model would label all the words included in the propagandistic span. In the second experiment, we examine **sentence classification** problem with built models. The models classify sentences as propagandistic or non-propagandistic.

Using the experiments mentioned above, we examine and evaluate the performance of the stacked system vs the standalone one. Our results (see Section 5.) identify the differences in approach to dealing with the detection of propagandistic fragments.

## 5. Results

Before incorporating classifier in the stacked system, we constructed two baselines to measure the classifier’s advantage on the sentence classification task. As a first baseline, we used AdaBoost (Schapire, 2013) with 200 estimators. For a second baseline, we chose Random Forest classifier (Breiman, 2001) with 80 estimators. We performed 10-fold cross-validation where 25% of training data was used for validation (i.e. train 67.5% - validation 22.5% - test

Table 2: F1-scores for different classifiers in sentence classification task

Model	Mean
BiLSTM	<b>0.5023</b>
AdaBoost	0.1743
Random Forest	0.1938

10%). Mean-summary statistics of their performance on these folds is presented in Table 2. Our classifier outperforms two baselines on the significance level  $\alpha = 0.01$ .

For the **first experiment**, the F1-score is calculated on token-level, which means token labels were compared instead of comparing character spans. This is why our results are not directly comparable to SemEval 2020 task 11 scores. In the first experiment, we demonstrated the clear advantage of the standalone model over the stacked one (see Table 3.). Intuitively the stacked model, which filters out non-propagandistic sentences, could have an advantage over the standalone one. It turns out that by ruling out non-propagandistic sentences, the stacked model lost some useful information, while the standalone model exploited that information from non-propagandistic sentences. Classifier’s error propagates into the sequence labeling task causing lower performance of the stacked model. The mean F1-score of the standalone model for sequence labelling is shown in Table 3., which is satisfying for such a complex task. Matched-pair t-test on 5-fold cross-validation (train 80% - test 20%) was conducted to test the performance difference of both models. The null-hypothesis (models do not differ in performance) can be rejected on the significance level  $\alpha = 0.01$ .

Table 3: F1-scores on 5 folds for both experiments and both models.

Fold	Experiment I		Experiment II	
	Stacked	Standalone	Stacked	Standalone
1	0.1785	0.2541	0.4670	0.5449
2	0.2071	0.2495	0.5298	0.5336
3	0.1370	0.2528	0.5499	0.5457
4	0.1605	0.2276	0.5411	0.5603
5	0.1221	0.2700	0.5323	0.5346
Mean	0.1610	<b>0.2508</b>	0.5241	0.5438

In the **second experiment**, our goal was to see if the standalone model is better at detecting propagandistic content

<sup>2</sup>(Chollet and others, 2015)

<sup>3</sup>www.spacy.io

The demand for an FBI probe is "utter nonsense," former U.S. Attorney Joseph diGenova said earlier this week.

The Democrat leadership collusion with Muslim spies is the biggest story of treason and espionage in the recent memory.

Figure 3: Examples of fully correct and partially correct labelled sentence. Underlined parts indicate true propagandistic labels and highlighted parts indicate predicted propagandistic labels

in a sentence than the stacked system. Namely, we used the standalone model as a classifier in such a way that if it labeled at least one token as propagandistic, the whole sentence is labeled as propagandistic. We compared this with the performance of the sole classifier from the stacked model. We concluded that, even though a standalone model outperforms a stacked model on sequence labelling task, it does not have a statistically significant advantage in performance on this classification task. As in the first experiment, a matched-pair t-test was conducted on 5-fold cross-validation to test the performance difference of both models. The null-hypothesis (models do not differ in performance) can not be rejected on significance level of  $\alpha = 0.05$ . The example of labelled sentences are presented in Figure 3.

## 6. Conclusion

In this paper, we explored different ways of building a model to detect propagandistic content in a sentence. We show that the standalone recurrent neural network model yields significantly better results than the multi-component model on sentence-level propaganda fragment identification task and we achieve satisfying results given the complexity of the problem.

In the future work we would like to examine and compare performances of attention-based models (BERT etc.) with our standalone RNN architecture. We would also try to incorporate contextual word embeddings in our model, which could ultimately lead to better results.

## References

Alberto Barrón-Cedeno, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019. Proppy: A system to unmask propaganda in online news. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9847–9848.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

François Chollet et al. 2015. keras.

Giovanni Da San Martino, Alberto Barron-Cedeno, and Preslav Nakov. 2019a. Findings of the nlp4if-2019 shared task on fine-grained propaganda detection. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 162–170.

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeno, Rostislav Petrov, and Preslav Nakov. 2019b. Fine-grained analysis of propaganda in news articles. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, EMNLP-IJCNLP 2019, Hong Kong, China, November.

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

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

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeno, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news articles. *arXiv preprint arXiv:1910.02517*.

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

Robert E Schapire. 2013. Explaining adaboost. In *Empirical inference*, pages 37–52. Springer.

# Bot Detection From a Single Tweet

Ivan Anić, Ivan Bilić, Silvije Škudar

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia  
{ivan.anic,ivan.bilic2,silvije.skudar}@fer.hr

## Abstract

With the advent of automatic content generation, it is becoming increasingly hard to tell whether information was crafted by artificial intelligence or real people. Even though it's always been possible to create harmful content, bots potentiate that capability significantly, and so the task of classifying content as generated by bots becomes vitally important, especially on sites used by many people. We focus on detecting bot-generated content on Twitter, a popular social media site. While most of the previous work combines tweets with account-level metadata, we focus on detecting bots solely from a single tweet's content. We experiment with two main approaches: a shallow classification model with handcrafted features and a deep model that exploits Twitter GloVe embeddings and BERT sentence embeddings.

## 1. Introduction

Social media has rapidly become an integral part of our lives, and a trusted resource of content. People are used to browsing and absorbing information from social networks on a daily basis. However, by trusting them too much, they can easily be influenced without even being aware. The power of social media bots lies in this subtlety of influencing public opinion, which can be used in harmful ways, such as spreading propaganda, fake news (Shu et al., 2018), increasing exposure to inflammatory content (Stella et al., 2018) and falsehood in general, political campaigns, etc.

Academic studies estimate that up to 15% of Twitter users are automated bot accounts (Varol et al., 2017), which is why the task of detecting bots on social networks has gained significant attention over the past decade. What makes a detection task even harder is that bots don't necessarily need to be fully autonomous, as they can also be supervised by a human. Human supervisors can introduce more variety into bot behaviour mechanisms, decreasing the chance for successful detection. Furthermore, bot creators can also analyze the research being done on the topic, together with the code being published, in order to improve their bots and increase their chances to remain "under the radar". It is for this reason that the ability to recognize bot-generated content merely from a single tweet's content is important (Ferrara et al., 2014).

## 2. Related Work

Detecting bots has been researched extensively, especially since the recent 2016 United States presidential election. Much of the work used the Twitter API<sup>1</sup> and combined tweet content with various user-level metadata. *BotOrNot* (Davis et al., 2016) used Random Forests with more than 1,000 features derived from the content itself as well as metadata such as account creation time, friend count, user's geographical and language information, all obtained using the Twitter API. Cai et al. (2017) use a deep network and combine text statistics with tweet timestamps and user's

tweet history. Similarly, for their deep model, Kudugunta and Ferrara (2018) also use a dozen account-level features like followers, friends and favourites counts. They also check for background images and default user settings. Dickerson et al. (2014) combine user metadata with tweet and in-depth sentiment analysis.

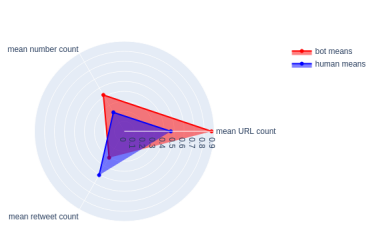
While behavioural data is powerful and has been studied extensively (e.g. Guess et al. (2019)), APIs for obtaining metadata are often inaccessible or limited. Thus, a different, albeit more difficult approach is one where only tweet content is used. Clark et al. (2015) use a classifier and analyze the average URL count per tweet, average pairwise lexical dissimilarity between tweets as well as the word-introduction rate decay over time-ordered tweets. More recently, Oliveira et al. (2019) combine tweet sentiment with tweet characteristic frequencies like URL and mention frequencies.

In a similar vein, our work focuses on classifying individual tweets with no additional metadata. We investigate the impact of features obtained from raw tweet text on performance of a shallow SVM model. Additionally, we tried generating entire tweet embeddings using a BERT (Reimers and Gurevych, 2019) based model, and after noticing significant differences between human and bot embeddings, we passed it through a fully connected layer to see whether this information could yield improvements, described in detail in Section 4.1.

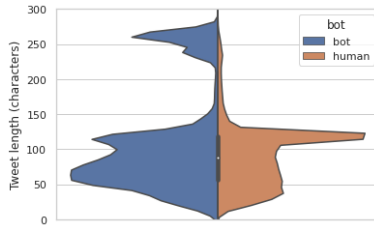
## 3. Dataset

Our dataset consists of 6760 accounts with 100 tweets per account written in English, which totals to 676000 tweets, where 264000 tweets are used for testing, and the remaining 412000 tweets for training. Additionally, of those 412000 tweets, 82400 tweets are used for validation when training a deep model. All the datasets have balanced class distributions. The dataset we used was published for PAN @ CLEF 2019 - Bots and Gender Profiling competition task (Rangel and Rosso, 2019). There is also a Spanish subset of the dataset and all the accounts in the dataset contain gender labels, but this paper is focused only on the bot de-

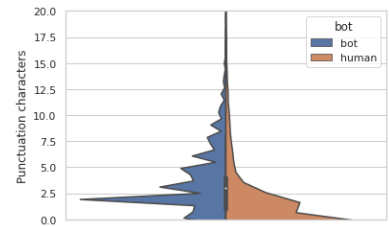
<sup>1</sup><https://developer.twitter.com/en>



(a) Comparison of mean URL, retweet and number counts,



(b) Character count distributions,



(c) Punctuation count distributions.

Figure 1: Comparison of feature values that significantly differed between bot and human content.

tection task using only tweets written in English.

### 3.1. Preprocessing

Special tokens such as URLs and numbers are replaced with special values. Stopwords, non-words and single characters are removed, HTML tags are stripped, text is lower-cased and so forth. Other than that, the way tweets are pre-processed differs across models. When using GloVe Twitter embeddings (Pennington et al., 2014), tweets are pre-processed using a slightly modified Python version of the Ruby script given on the GloVe official website, and tokenized using NLTK TweetTokenizer. For BERT embeddings, SpaCy’s rule-based sentence segmentizer is used to extract the sentences from tweets.

## 4. Our Approach

We experiment with different model approaches. First, we use a baseline SVM model that takes handcrafted features as input. Handcrafted features are chosen based on observations described in Section 4.1. Next, we employ two different deep models. The first one takes GloVe word vectors pretrained specifically on a Twitter corpus as input and it’s composed of LSTM and fully connected output layers. How we use GloVe embeddings is pointed out in Section 4.2. The second model is a sentence transformer that is used to produce and feed tweet embeddings into the fully connected layers. Sentence transformer is not fine-tuned. Section 4.3. explains tweet embeddings.

### 4.1. Handcrafted Features

To gain a better understanding of the differences between bot and human behaviour, we crafted new features from tweet content. For each tweet, 16 statistical features are created. These can be roughly grouped in four categories: word features (WF, e.g. word count, character count), punctuation features (PF, e.g. punctuation count), unique word features (UF, e.g. number of unique words) and special token features (SF, number of retweets, URLs and numbers). Tweets were separated in two groups, one containing bot tweets, and the other containing human ones. This lets us compare the mean and standard deviation of each feature’s values per tweet.

We analyzed each feature separately, and while some differed significantly, many were redundant. Interestingly, Figure 1a shows that bots posted URLs and numbers almost twice more than human users on average, while people retweeted more. Since the dataset is of political nature,

higher URL count could be explained by suspecting these posts linked to external political or propaganda sites. Additionally, it seems a number of bots pushed the tweet length near its 280 character maximum, while human users usually remained below 150 characters per tweet, as shown on Figure 1b. From Figure 1c it’s visible that there was also more variance in punctuation use in bot content.

Keeping this insight in mind, we scaled the new features so they could later be used as inputs to the SVM model. Even though we use all 16 features as inputs to the model, we do note that some of them might be redundant, as they do not differ significantly between the bot and human classes.

### 4.2. Word Embeddings

We train our deep model on pretrained GloVe Twitter word vectors (Pennington et al., 2014). The tweet preprocessing done in this approach is briefly explained in Section 3.1.. For each tweet, all the tokens are replaced with their corresponding GloVe Twitter embedding. Tokens that have no corresponding embedding are replaced with a random embedding sampled from gaussian distribution with zero mean and unit variance.

### 4.3. Tweet Embeddings

Parameter estimation on our dataset gives results that motivate the usage of tweet embeddings. Each user is represented with a particular multivariate gaussian distribution and each tweet embedding is then viewed as a vector sampled from its user’s distribution. We can estimate user’s variance vector  $\sigma$ , calculated as  $diag(\Sigma)$  where  $\Sigma$  is a covariance matrix describing the user’s distribution. Calculating the mean variance vector for humans and bots separately, and comparing the two vectors yields that humans have higher variance in 742 out of 768 dimensions, meaning that tweet embeddings capture the fact that human behaviour is less predictable. However, the only use of that statistical analysis is to encourage, or at least not discourage the use of tweet-level embeddings, meaning we do not embed the variance information into the models in any way. Regarding statistical significance of those results, t-test results show that for 730 out of 768 embedding dimensions, human tweet embeddings differ from bot generated ones at a significance level of 5%, counteracting the potential occurrence of a false positive by implementing Bonferroni correction (Bonferroni, 1935).

In this approach, we generate tweet-level embeddings using a pretrained sentence transformer based on BERT-base

Table 1: Classification accuracy and macro F1-score for the handcrafted feature experiment, GloVe + LSTM experiment and BERT experiment.

Model	Accuracy	Macro F1-score
TFIDF + SVM	0.755	0.752
TFIDF + SVM + WF	0.760	0.757
TFIDF + SVM + PF	0.757	0.754
TFIDF + SVM + UF	0.763	0.760
TFIDF + SVM + SF	0.754	0.751
TFIDF + SVM + ALL	<b>0.770</b>	<b>0.768</b>
GloVe + LSTM	0.806	0.788
BERT	<b>0.812</b>	<b>0.792</b>

model<sup>2</sup>. Given a sentence, sentence transformer generates a 768-dimensional sentence embedding. Tweet embedding is then constructed by averaging all sentence embeddings that belong to the same tweet. The sentence transformer is not fine-tuned on our data. In contrast to GloVe, tweet embeddings are generated for each tweet in our dataset so in this case there’s no need to use the random gaussian embedding.

## 5. Experiments and Results

Having acquired an idea about the difference between content generated by bots and people, we conduct an experiment to empirically show whether extracting statistical information from raw tweet content improves bot detection. To this end, we separate handcrafted features into four groups as described in Section 4.1.. Our baseline consists of sklearn’s unigram TFIDF Vectorizer to convert preprocessed tweets into 10,000 dimensional vectors as inputs for the SVM. We then experiment with concatenating each of the four feature groups to obtain TFIDF vectors separately, as well as combining all of them. SVM is then trained on the entire train dataset and evaluated on the test dataset. The results of these experiments are shown in the upper section of Table 1. Surprisingly, special token features (SF) do not improve the scores despite their frequencies being significantly different, as shown in Figure 1a. However, the combination of all handcrafted features does improve the scores somewhat.

In GloVe + LSTM experiment, first the text is preprocessed in order to be coherent with the GloVe Twitter pre-trained word vectors. Those embeddings are then fed into LSTM whose output is fed to fully connected layers. More about the preprocessing steps and word embeddings can be found in Section 3.1. and Section 4.2. respectively. We use LSTM with four layers. Furthermore, in all experiments except the baseline, we use two fully connected layers. The first one with ReLu and the second, output layer with the Sigmoid activation function which outputs the probability of an input tweet being written by a bot. We observe that for training with different GloVe embedding sizes, changes in performance are negligible.

BERT experiment is essentially employing a BERT-based sentence transformer that outputs 768-dimensional sentence embeddings from which tweet embeddings are then constructed. Tweet embeddings are then fed into fully connected layers that are the same as in GloVe + LSTM experiment. See Section 4.3. about tweet embeddings. The results in Table 1 show that BERT embeddings are dominant in bot detection task as well. GloVe embeddings need to be processed by LSTM in order to come close to BERT’s performance, whose embeddings are directly fed into the fully connected layers, with no additional processing in between.

## 6. Conclusion

In this paper we presented an overview of the task of detecting social media bots, and proposed a few approaches to classify bots on Twitter based solely on the tweet contents. Although the information to make conclusions upon is in this way very limited, we have successfully shown that text generated by bots is still different enough than that of humans to successfully distinguish bots from humans.

There are several promising directions for future improvements on proposed methodologies, such as experimenting with different network architectures, hyperparameters, as well as combining handcrafted features with word embeddings. Regarding BERT, word embeddings could be used instead of sentence embeddings, and BERT model fine-tuning could also be carried out during training.

## References

- Carlo E Bonferroni. 1935. Il calcolo delle assicurazioni su gruppi di teste. *Studi in onore del professore salvatore ortu carboni*, pages 13–60.
- C. Cai, L. Li, and D. Zengi. 2017. Behavior enhanced deep bot detection in social media. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 128–130.
- Eric M. Clark, Jake Ryland Williams, Richard A. Galbraith, Chris A. Jones, Christopher M. Danforth, and Peter Sheridan Dodds. 2015. Sifting robotic from organic text: A natural language approach for detecting automation on twitter. *CoRR*, abs/1505.04342.
- Clayton Allen Davis, Onur Varol, Emilio Ferrara, Alessandro Flammini, and Filippo Menczer. 2016. Botornot: A system to evaluate social bots. In *Proceedings of the 25th International Conference Companion on World Wide Web, WWW ’16 Companion*, page 273–274, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.
- John P. Dickerson, Vadim Kagan, and V. S. Subrahmanian. 2014. Using sentiment to detect bots on twitter: Are humans more opinionated than bots? In *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM ’14*, page 620–627. IEEE Press.
- Emilio Ferrara, Onur Varol, Clayton A. Davis, Filippo Menczer, and Alessandro Flammini. 2014. The rise of social bots. *CoRR*, abs/1407.5225.
- Andrew Guess, Jonathan Nagler, and Joshua Tucker. 2019. Less than you think: Prevalence and predictors of fake

<sup>2</sup><https://github.com/UKPLab/sentence-transformers>

- news dissemination on facebook. *Science Advances*, 5(1).
- Sneha Kudugunta and Emilio Ferrara. 2018. Deep neural networks for bot detection. *CoRR*, abs/1802.04289.
- Rodrigo Ribeiro Oliveira, Cláudio Moisés Valiense de Andrade, José Solenir Lima, Conceição Silva, and Almir Moreira da Silva Neto. 2019. Bot and gender identification: Textual analysis of tweets.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Francisco Rangel and Paolo Rosso. 2019. Pan19 author profiling: Bots and gender profiling, February.
- Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11.
- Kai Shu, Suhang Wang, and Huan Liu. 2018. Understanding user profiles on social media for fake news detection. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 430–435. IEEE.
- Massimo Stella, Emilio Ferrara, and Manlio De Domenico. 2018. Bots increase exposure to negative and inflammatory content in online social systems. *Proceedings of the National Academy of Sciences*, 115(49):12435–12440.
- Onur Varol, Emilio Ferrara, Clayton A. Davis, Filippo Menczer, and Alessandro Flammini. 2017. Online human-bot interactions: Detection, estimation, and characterization. *CoRR*, abs/1703.03107.

# To Context or Not to Context? Analysis of Non-contextualized Word Embeddings in Propaganda Detection Task

Luka Barišić, Marko Pisačić, Filip Radović

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia  
{luka.barisic, marko.pisacic, filip.radovic}@fer.hr

## Abstract

In this paper, we explore the impact of non-contextualized word embeddings on the sequence labeling task of propaganda fragment identification. More specifically, we evaluate the addition of non-contextualized word embeddings to contextualized ones, testing the hypothesis that they will give additional, more general information to the model. The results do not show an improvement of statistical significance, however, we discuss potential reasons behind them, and propose further research to analyze this problem more deeply.

## 1. Introduction

Propaganda is a mode of communication used to manipulate or influence the opinion of groups to support a particular cause or belief. Though its use is not exclusively negative, propaganda very often involves a heavy emphasis on the benefits and virtues of one idea or group, while simultaneously distorting the truth or suppressing the counter-argument.

With the dawn of the internet, propaganda has become very influential and a part of our daily lives. Propaganda is usually written in a way that can't be detected by the reader. That can become dangerous because it can shift the reader's perception without the reader knowing it. Because of its huge impact and the danger it poses to our lives, much effort has been put into stopping and detecting propaganda.

One limitation of current approaches to propaganda detection is that they use limited knowledge about outside, more general information. That poses a problem because propaganda often subtly changes the meaning of the words in a sentence. By not knowing the meaning outside that sentence, we deprive ourselves of information.

Our idea is to construct general word representations that may bring additional information about the base meaning of the word as a contrast to the meaning of the word in a particular context. For example, consider the propagandist sentence: "Entering this war will make us have a better future in our country". We expect the usually negative connotation of the word 'war' to be alleviated by its context, more precisely by the phrase 'better future'. Our assumption is that this would be reflected in the word embeddings, so the contextualized and non-contextualized embeddings would be different. We expect that this difference will behave in a more specific way when dealing with embeddings of a propaganda fragment. Our model should then be able to capture those specifics and be better informed which would result in a better performance.

In section 2., we give an overview of the work done on propaganda detection so far. We evaluate our new embeddings on the task of fragment-level propaganda detection. Our model is explained in section 3. We also elaborate on how certain layers of BERT behave and the type of information they encode. Although our results don't show any

improvements over using only standard embeddings, we argue why the concept should be further explored.

## 2. Propaganda Detection in News Articles

Recently, there has been a lot of interest centered around detecting propaganda. There are multiple types of propaganda detection tasks, but they can all be approached in similar ways. Barrón-Cedeño et al. (2019) showed that modeling writing style and text complexity are very effective in detecting propaganda on the article level. They argue that articles that contain propaganda have different writing styles compared to normal articles. With the release of powerful pre-trained models like BERT and ELMO, researchers have tried to tackle more fine-grained tasks of propaganda detection. Da San Martino et al. (2019) released a corpus that contains sentence- and fragment-level labels. They proposed a novel multi-granularity neural network which was trained on both sentence- and fragment-level data and achieved state of the art results. By solving two problems at the same time, the model was able to combine the knowledge that was learned from both tasks. Yoosuf and Yang (2019) used a pre-trained BERT model and fine-tuned it on the fragment-level classification task. They also observed that solving the class imbalance problem improved their model drastically. We will see that our model also takes the problem of imbalance into consideration. Hou and Chen (2019) used article titles as additional context for their BERT model. They argued that the titles could serve as a summary of the whole article thus provide additional knowledge to the model.

Similar to propaganda detection, there has also been a new wave of research centering around analyzing and interpreting state-of-the-art natural language processing models. Tenney et al. (2019) show that specific parts of BERT layers encode different information. They reported that basic, syntactic information appears in the lower layers, while more abstract, semantic information appears at higher layers. We investigated those differences in layers and based our model on the results which will be discussed in a future section.

Propaganda detection is a complicated problem that doesn't have one clear solution. Although previous work

Table 1: Vector similarities of the word killed compared to the meaning in the sentence "Someone was killed in a horrifying manner". Cosine refers to cosine similarity, and Euclid to Euclidean distance

Text	Cosine	Euclid
He brutally killed someone	0.7988	43.771
He accidentally killed someone	0.7915	44.649
He killed someone	0.7995	43.830

is fairly successful, all the methods mentioned above are too focused on only extracting information from the corpora they are dealing with. We argue that adding a more general representation of the text information can add better insights to the model.

### 3. Experimental Setup

#### 3.1. Dataset

The data for our task is supplied by the Propaganda Analysis Project <sup>1</sup>, as a part of SemEval 2020 shared task 11: "Detection of propaganda techniques in news articles". The task is split into two parts: identification of text fragments containing propaganda, and classification of these fragments into one of 18 specific propaganda techniques, such as whataboutism, red herring, and name-calling. We chose to focus only on the first part — propaganda fragment identification, as we found it more interesting to determine what differentiates propaganda from non-propaganda, rather than improving on the well-researched multi-class classification problem.

Since the mentioned task was active during the implementation of our model, we were unable to obtain the official test set, so we decided to combine train and validation sets into one dataset, and then did our own train-validation-test splits. The dataset consists of 445 news articles with manually annotated text fragments containing a propaganda technique. There are 19 830 sentences in total, which we split into tokens with a simple tokenizer. Out of a total number of 401,705 tokens, only 51,265 (12,8%) of them were part of fragments labeled as propaganda, and the remaining 350 440 (87,2%) as non-propaganda.

#### 3.2. Embedding Creation

Input tokens (words, sub-words, and other parts of a sentence) are represented as numeric vectors called word embeddings. An embedding for a word can either be dependent on the context of the word (dynamically generated), or not (static). As discussed in Section 2., we would like to explore the impact of adding non-contextualized word embeddings to our baseline model, which is using contextualized embeddings only.

To obtain non-contextualized embeddings for all words in our corpus, we used the "All the news" dataset provided by Andrew Thompson.<sup>2</sup> First, we found all occurrences of a certain word in the "All the news" dataset. We then

<sup>1</sup><https://propaganda.qcri.org/index.html>

<sup>2</sup><https://www.kaggle.com/snappcrack/all-the-news>

averaged the contextualized BERT embeddings of all those occurrences to obtain the non-contextual embedding. The idea is that by considering all possible contexts of the word, the contextual information fades away and all that remains is the non-contextual meaning of the word.

The obtained non-contextual embeddings are concatenated with contextualized BERT embeddings, and fed into our model as input. We expect the model to perform better than without non-contextualized embeddings. To provide further insights we also consider two simpler models which we expect to perform worse than both the baseline and our model. The first one uses non-contextualized embeddings only, and the second one uses another type of well-known static embeddings – GloVe.

#### 3.3. Embedding Exploration

In this subsection, we will do an analysis of created non-contextual embeddings. To better understand how the embeddings compare to the contextual embeddings, we compared how similar are the contextual and non-contextual embeddings of the same word. The average cosine distance between those two embeddings in every BERT layer is shown in Figure 1. We can observe that the embeddings are most distant at higher layers of the BERT architecture, which guided our decision to use the average of only the four top-most layers in both our contextualized and non-contextualized embeddings. We further observed that the embedding similarities are different based on the contexts the words are in. In Table 1 we can see how this plays out with some real-world example sentences.

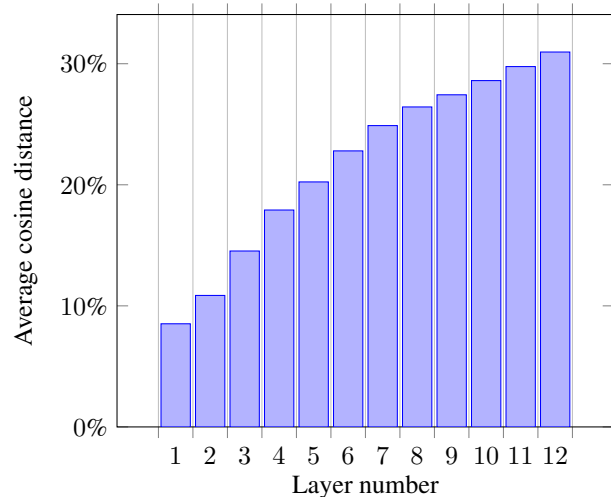


Figure 1: Average cosine distance between contextualized and non-contextualized embedding for each BERT layer

#### 3.4. Model Architecture

The architecture of the model can be seen in Figure 2. Since our features are comprised of concatenated word embeddings, on the entrance of our model there are two word encoders — BERT, and the non-contextualized custom encoder, described in a previous subsection. BERT (Bidirectional Encoder Representations from Transformers) is a

transformer model which uses attention mechanism to learn contextual relation between words (or sub-words) in a sequence. In that way, for each input token, BERT analyses its context and produces a dynamic embedding. We use that output as the first half of our feature vector. The second half is the output of the second, custom encoder, which in this case performs a simple lookup function for every token.

After the embeddings of the token are obtained, they are sent to the bi-directional LSTM (BLSTM). We chose this variant of a recurrent neural network because of its ability to capture time dynamics in sequences, and has proved effective in other sequence labeling tasks. Before the BLSTM, there is a fully-connected linear layer with an equal number of inputs and outputs. After the BLSTM, there is also a linear layer with two outputs decoding the probabilities of non-propaganda and propaganda classes for that token. Optimization of the network parameters is done using stochastic gradient descent, with a cross-entropy loss function.

Hyperparameters for this model can be found in Table 2. Because of the lack of resources to properly train all models, we optimized hyperparameters only on one of the baselines and used them for all models. The optimal hyperparameter values for that baseline were found using a simple grid-search optimizer.

### 3.5. Evaluation

The main evaluation metric for our problem is a custom, task-adjusted F1 score used in the SemEval on which we base our paper. The gist of the metric is that it deals with normalized overlaps of fragments  $s$  and  $t$  given by the function  $C$ :

$$C(s, t, h) = \frac{|s \cap t|}{h} \quad (1)$$

where  $h$  is the normalizing factor. We can think of  $C$  being the precision on an individual predicted propaganda fragment if the normalizing factor is  $|s|$ , or recall on an individual reference propaganda fragment if the normalizing factor is  $|t|$ . Overall precision is then calculated by averaging precisions on individual predicted fragments. Calculation is analogous for overall recall. For a set of predicted fragments  $S$  and a set of reference fragments  $T$  the formulae for overall precision and recall are:

$$P(S, T) = \frac{1}{|S|} \sum_{\substack{s \in S \\ t \in T}} C(s, t, |s|)$$

$$R(S, T) = \frac{1}{|T|} \sum_{\substack{s \in S \\ t \in T}} C(s, t, |t|)$$

The main evaluation is F1 metric, the usual harmonic mean of the overall precision and recall.

## 4. Results

The training and evaluation of models with each of the four combinations of word embeddings was done on 10 different train-validation-test splits. Each dataset split was fixed during the training of all four models so the results for the split are directly comparable. Our null hypothesis is that the

Table 2: Hyperparameters used during model training.

Hyperparameter	Value
BLSTM state size	200
Dropout	0.2
Initial learning rate	0.05
Annealing factor	0.2
Patience	1
Epochs	5
Batch size	10

Table 3: Mean values for precision, recall, and F1 score for each model. Combined model is our proposed model, which uses both non-contextualized and BERT embeddings. Sample size  $N = 10$

Model	Precision	Recall	F1
GloVe	44.83	8.06	13.31
Non-contextualized	30.95	29.65	29.18
BERT	33.11	49.91	38.47
Combined model	36.36	43.13	38.76

results with the contextual embeddings alone are at least as good as the results with the added non-contextual embeddings. We would like to see our null hypothesis rejected with the significance level  $\alpha = 0.05$ . That way we could conclude that the addition of non-contextual embeddings improved the performance of our model with a statistical significance.

The mean values for the contest specific precision, recall and F1 measures for each embedding combination are given in Table 3. Although we observe both a better mean and pair-wise performance of our model compared to the baseline, the t-test was performed which showed that, unfortunately, the improvement is not statistically significant ( $p = 0.351 > \alpha$ ). As was expected, the contextualized-embedding baseline outperforms both non-contextualized and GloVe embeddings on their own ( $p < 10^{-4}$ ).

We can assume a couple of reasons for the lack of statistically significant improvement in performance. Since hyperparameters were optimized on a baseline which included only contextualized embeddings, it seems probable that the doubling of input feature size has caused our model performance to suffer. Double input size could increase the capacity our model needs to have, and hence demands additional tweaks to our model and hyperparameters. Also, our custom non-contextualized word embeddings aren't ideal. What we tried to do was to capture only the basic meaning of a word in a particular sentence. What was actually done was averaging across word occurrences in a big dataset. In that way, the most frequent word meaning dominates the embedding and isn't tailored in any way to the meaning in a specific sentence. We also weren't able to fine-tune BERT layers due to computational limitations.

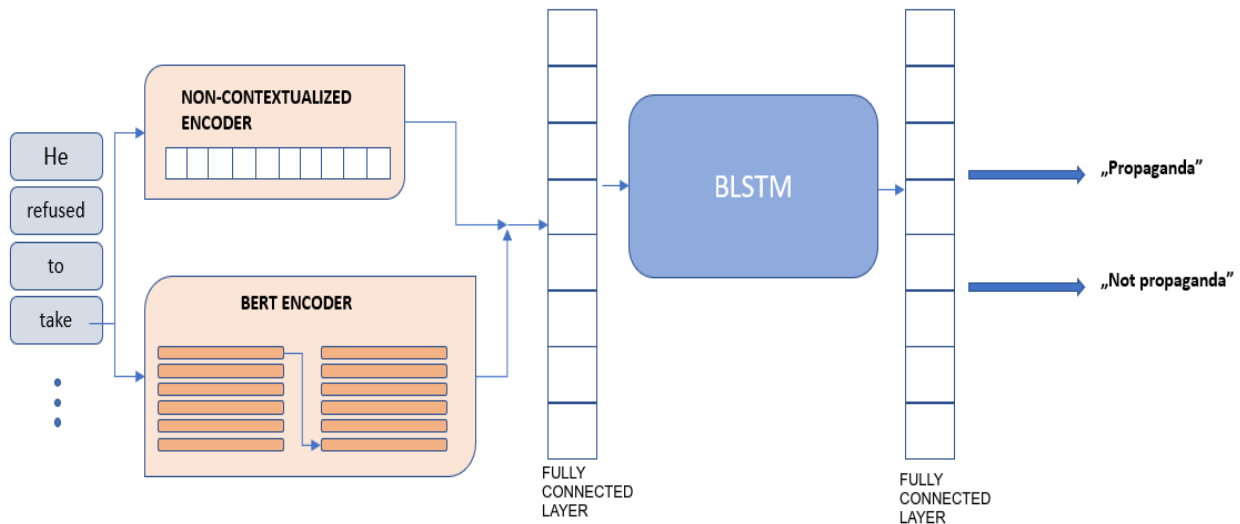


Figure 2: Model architecture

## 5. Conclusion

This paper investigated the task of propaganda detection, and described it as a sequence labeling task. We explored the role of non-contextualized embeddings in a propaganda fragment identification task. We proposed a way to construct custom, non-contextualized BERT embeddings and evaluated their impact on the performance of sequence labeling. Our hypothesis was that propaganda subtly changes the meaning of words and that by utilizing non-contextualized embeddings the model can detect those differences. Our embedding exploration confirms that changes in context can be detected using BERT embeddings. Although our idea initially looked promising, our results were inconclusive and we didn't manage to outperform our hard baseline that used only contextualized embeddings. We argue that the fault may lie in our custom non-contextualized embeddings which were too static and couldn't capture the base meaning of the word in a sentence well. We were also limited by computational resources, so the model training could certainly be improved in further work. Even though we didn't reach any conclusive support for our hypothesis, we believe that non-contextualized embeddings could have their role in propaganda detection. In our future work, we will test how other non-contextual embeddings work with BERT embeddings. We will also experiment with more model architectures that can better capture the underlying structure of our data.

## References

- Alberto Barrón-Cedeño, Israa Jaradat, Giovanni [Da San Martino], and Preslav Nakov. 2019. Propy: Organizing the news based on their propagandistic content. *Information Processing Management*, 56(5):1849 – 1864.
- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news article. In

*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China, November. Association for Computational Linguistics.

- Wenjun Hou and Ying Chen. 2019. CAUnLP at NLP4IF 2019 shared task: Context-dependent BERT for sentence-level propaganda detection. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 83–86, Hong Kong, China, November. Association for Computational Linguistics.
- Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. *CoRR*, abs/1905.05950.
- Shehel Yoosuf and Yin Yang. 2019. Fine-grained propaganda detection with fine-tuned BERT. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 87–91, Hong Kong, China, November. Association for Computational Linguistics.

# Using Additional Features in Emoji Prediction

Bruno Belić, Simon Grgurina, Barbara Lugar

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia  
{bruno.belic, simon.grgurina, barbara.lugar}@fer.hr

## Abstract

Emojis are a constant presence in online communication and their presence gives additional information to text. Because of that, the task of creating the best emoji prediction model has been explored before. As a follow up research on the SemEval 2018 Task 2, this paper explores the impact of 8 handcrafted Twitter writing style features in the task of emoji prediction.

## 1. Introduction

Emojis' popularity has been on the rise since the late 2000s when the rapid growth of social media popularity occurred. Social media users often use emojis as a way of expressing emotion, sentiment, or just adding a visual representation of what the text is about. Therefore, analyzing social media textual data should always include an emoji analysis because they contain valuable information that should never be overlooked. The task of emoji prediction has become a popular one since efficient emoji prediction could be used in gaining better results in some other NLP tasks, such as emoji suggestions or information retrieval (Barbieri et al., 2017).

The task has proven to be a difficult one to solve, not only by machine learning models, but also by humans (Barbieri et al., 2018a). The problem lays in the fact that there are many emojis that fall into the same category, especially the ones expressing emotions, and sometimes picking one is just a matter of personal preference.

This paper explores whether some text features can help improve emoji prediction. Textual data created from social media platforms is often more than just words. Language concepts such as the use of caps lock, letter repetition in words and multiple exclamation-marks give different meanings to words and tweets. Our goal is to explore the correlation of these, and similar features, with the emoji label and whether these features could be used in gaining better performance of our models for emoji prediction.

## 2. Related Work

The work on emoji classification has its roots in predicting the most likely emoji from a Twitter message (Barbieri et al., 2017) using an LSTM-based model. Work has also been done on other platforms such as predicting emojis on a mobile keyboard (Ramaswamy et al., 2019) and emoji prediction for Instagram posts by combining text and picture information (Barbieri et al., 2018a).

The SemEval Task 2 (Barbieri et al., 2018b), an emoji prediction competition for English and Spanish language Twitter data, created a considerable amount of work. The goal was to create the best performing model for predicting the correct emoji for Tweets, based on its text.

In this paper we use a system based on a BiLSTM, much like several top-performing competition systems (Baziotis

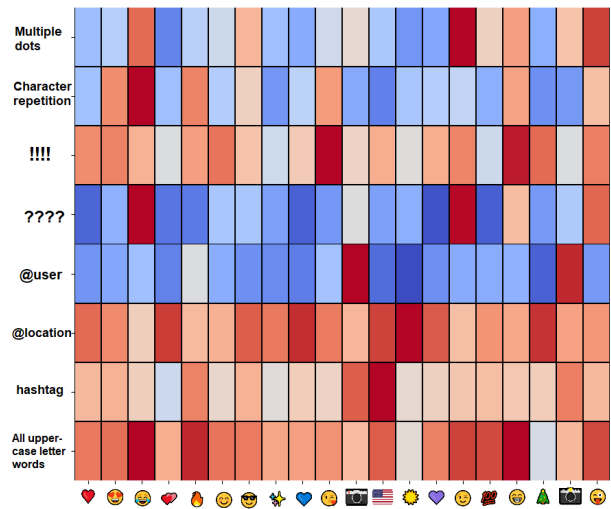


Figure 1: Figure shows relative frequency of custom features for each class (emoji). Darker red means higher relative frequency, while lighter blue means lower frequency. The coloring is relative to the maximum relative frequency for the feature (maximum row value).

et al., 2018; Liu, 2018). Unlike the competitors our goal is not to create a state of the art system for emoji classification. Instead, we explore whether additional feature information will improve emoji prediction.

## 3. Dataset

The data used in this work is data from the SemEval Task 2 competition (Barbieri et al., 2018b). While the data contains both English and Spanish dataset, only the English dataset was used in this work and all further writing will refer to it. The data consists of text from Tweets which contain exactly one emoji, and the emojis as the labels. Only tweets which contain the 20 most used emojis appear in the dataset. The tweets were gathered between October 2015 and January 2018, and were geolocalized in the USA. The data is split in trial and test sets of 50 000 tweets each, available directly, and the training set which is downloaded using the Twitter API. Because of tweet unavailability only 428 260 tweets were retrievable, of the 500 000 tweets which originally comprised the training set. Due to

❤️	😄	😓	❤️	🔥	😊	😁	🌟	💙	😬	📷	us	☀️	💜	😏	👄	🤔	🎄	📷	😬
92855	45030	44338	22746	21692	20207	18697	15660	14386	14025	14245	13137	11793	10995	11788	11662	11485	11094	11685	10740
46352	22650	22106	11303	10774	10168	9287	7800	7184	6989	7013	6680	5864	5563	5911	5824	5711	5584	5933	5434

Figure 2: The 20 emojis used in the English dataset and their counts before and after halving the dataset

the lack of resources only half of the training dataset was used. The emojis and their respective counts in the training dataset before and after it was split in half can be seen on Figure 2.

#### 4. Feature Extraction

We constructed a set of custom features to further explore the effect of certain writing style features that we consider a byproduct of modern means of communication, such as social media and Twitter. Textual data found on Twitter and similar websites is often grammatically incorrect or is intentionally injected with emotion through letter repetition, use of upper-case letters, or even special character repetition, e.g. exclamation marks. These features are usually intentionally added by the user to express excitement, anger, and other emotions. Emojis are used as an expression of emotion, which gives us a reason to believe the presence of these features could correlate with the emoji used in the tweet. To explore the effect of those features we created 8 features based on the Twitter text. Features count the presence of the following:

1. Multiple dots in a row
2. Letter repetition in words
3. Exclamation marks
4. Question marks
5. @user tags
6. @{some location} tags
7. Hashtags
8. Words written only in upper-case letters

For each tweet, a vector containing these counts is created. The features are extracted after tokenization, but before preprocessing using regexes. The result of the feature extraction step is a matrix of dimension  $N \times 8$ , where  $N$  is the number of tweets in the dataset and 8 is the number of custom features.

Figure 1 shows the presence of features throughout the tweets for each of the possible emojis. It can be noted from the figure that there are some features, such as presence of hashtags, location tags and use of upper-case letter words, that occur in tweets for all emoji labels, without many standing out. On the other side, there are some such as the use of ellipsis('...') that appear the most with a certain emoji, here it is the winking emoji. This figure supports our primary assumptions about these features being correlated with the use of emojis. The statistical significance of these features in accordance with the emoji label was proven using a chi-squared test with the level of significance set to 0.01.

#### 5. Preprocessing

We chose to preprocess the data by eliminating the custom features from the text, so we would be able to isolate the features and analyze their significance separately. Pre-processing steps for both models are removal of all special characters (@ and # included), case normalization and removal of '@user' tags. The text is tokenized with a standard English tokenizer.

We decided not to eliminate the character repetition as there is no unique formula for deciding whether to keep one or two letters. Even though its more frequent that the sequence of the same letters should be replaced with a single letter, there are also examples where this kind of letter substitution could damage our data. Consider the example of 'gooooo' appearing in a tweet, it is most likely that it's the word 'good' being extended for the purpose of emphasizing how good something is, but replacing the sequence of 'o' letters with a single 'o', we would end up with the word 'god', instead of 'good', which could be very misleading for the classifier, especially considering we are using a bag-of-words model for the baseline.

The character repetition presented a potential problem only for the baseline, while it did not present a problem for the main model as pretrained GloVe embeddings (Pennington et al., 2014) trained on a Twitter dataset are used. These pretrained embeddings account for intentional character repetition and misspelled words, chatting abbreviations and other forms of irregularities which are not found so often in other corpuses. These embeddings were an obvious choice since our dataset consists of Tweets which do not hold to the grammatical standards of the formal language and therefore can profit from embeddings trained on a suitable corpus.

#### 6. Methods

##### 6.1. Baseline

For the baseline model, we settled on logistic regression combined with a bag-of-words model for text representation. We have also included this baseline in the custom feature experiment to determine whether those features could enhance a shallow model's generalization ability in emoji prediction. Therefore, we trained our model on bag-of-words only (BOW-LR), and on bag-of-words combined with the set of custom features (BOW-LR+).

##### 6.2. Main Model

For our main model we used BiLSTM to process the embeddings in a sequence based manner and learn to extract features from the text. Output features from the BiLSTM

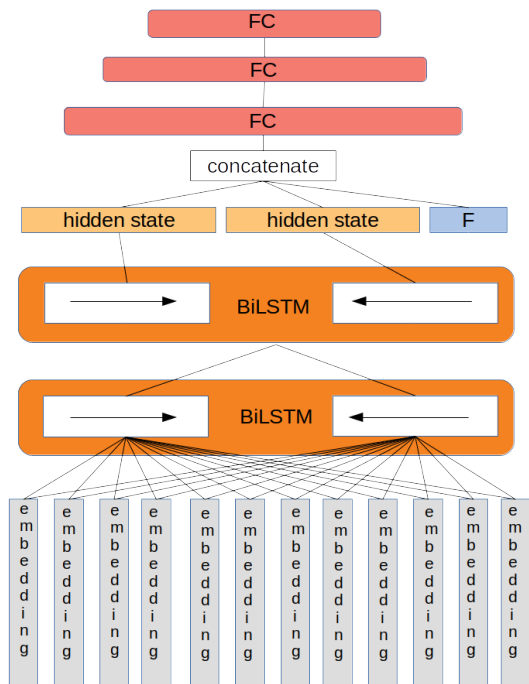


Figure 3: Figure shows the architecture of our main model. Number of embeddings is batch-specific and it does not correspond to the number in this figure. Light blue cell F represents the additional feature vector. Hidden state cells represent the last hidden state of each direction of the BiLSTM. Red FC cells represent fully connected layers.

are then concatenated with the features generated in the feature extractor. This concatenated vector is passed to fully connected layers to make the classification. This architecture is visible in Figure 3. This ability to customize the model and add our features in the middle of the processing instead of adding them at the beginning was also a large factor in choosing this architecture. Combining our features with features of already processed text seemed as a superior option than just concatenating them with a bag of words input (as we did in the baseline) or finding a way to include them in the embeddings.

ReLU is used as the activation between the layers, and cross entropy loss is used as the criterion function. Due to large class imbalance, we were unable to train the model to predict classes with smaller number of instances in the dataset. This problem was effectively solved by introducing weights to the criterion function. Weights were calculated by taking the inverse count of the instance and multiplying it with the count of the most frequent, so that the weight of the most common class is 1 and all the other weights are higher and inversely proportional to class counts.

Parameters of the model are quite basic, since fine tuning a deeper and more complicated model for better results

was not the point of this paper. 100-dimensional embeddings were used since they proved to yield better results than the more informational 200-dimensional embeddings, along with reducing the training time. BiLSTM is used because of its state of the art abilities sequence processing and powerful capacity in discovering context in NLP. Two layers of bi-directional LSTM are used with a hidden size set to 200. To reduce overfitting, a dropout of 0.5 is introduced between the BiLSTM layers.

As for the optimizer, stochastic gradient descent with a large learning rate of 0.1 was favoured over an Adam optimizer on which we experimented with smaller learning rates, ultimately without any improvement of the training process. In the training process, batches of size 32 were used. Number of epochs varied for the model with and without additional features. Model with additional features reached optimal results sooner (epoch five) than the model without features (epoch nine), possibly because it relied more on the fully connected layers which didn't have any dropout.

## 7. Evaluation

Finally, we report our results on the test set and discuss whether these features gave more insight to our data. Regarding the classification task itself, this twenty-class emoji classification is a difficult one. Between the twenty most common emojis, a number of them are quite similar and the main problem is most of them refer to some kind of positive sentiment. There are no sad or angry emojis, but there are however four different heart emojis (one of them is a double heart), two different camera emojis, a happy and a happier emoji, one that winks and shows his tongue and one that just winks. It is quite obvious how this prediction task would be hard even for humans.

For evaluation metrics, we decided to use the standard accuracy, recall, precision and F1-score. Since this is a multi-class prediction problem, we had to decide which type of average metric we will use. The *SemEval 2018 Task 2: Multilingual Emoji Prediction* competition holders recommended the use of the macro average over the weighted one because of the skewed distribution of the emoji class label (Barbieri et al., 2018b). Therefore, we decided to define the macro F1 as our main evaluation metric for model comparison.

Table 1: Evaluation metrics results on the test dataset.

Model	Accuracy	Precision	Recall	F1
BOW-LR	0.22	0.19	0.21	0.20
BOW-LR+	0.23	0.20	0.21	0.20
GloVe-BiLSTM	0.28	0.22	0.25	0.21
GloVe-BiLSTM+	0.31	0.23	0.25	0.22

### 7.1. Baseline

Baseline BOW-LR and BOW-LR+ performed very similarly on the task of emoji prediction with macro F1 and

recall being 20% and 21%, respectively, while on the other metrics there is a slight, but not a significant improvement. Even though we have proven the significance of our custom features, their inclusion did not improve the model's performance.

## 7.2. Main Model

Regarding the main model, it has outperformed the baseline, with accuracy being the most improved measure. When comparing the main model with additional features (GloVe-BiLSTM+) and without them (GloVe-BiLSTM), we have to conclude that there is no significant difference in any evaluation metric. Most improved measure is again accuracy, with other measures improving marginally and recall not improving at all.

## 8. Conclusion

Although additional features can improve a classifier, our features were unable to get significant improvements. This might be due to the large amount of information already contained in the GloVe embeddings trained on Twitter data, making these custom features mostly redundant. Main model in this paper is not well tuned and would most likely be able to perform better with more extensive fine tuning of the hyper-parameters. It, however, presents an approach to deep learning architectures where adding custom features to the ones obtained from the feature-extracting part of the deep learning model could boost its results. For future work, additional testing can be done, such as tf-idf scaling and normalization of the features or exploring a different set of features.

## References

- Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017. Are emojis predictable? *arXiv preprint arXiv:1702.07285*.
- Francesco Barbieri, Miguel Ballesteros, Francesco Ronzano, and Horacio Saggion. 2018a. Multimodal emoji prediction. *arXiv preprint arXiv:1803.02392*.
- Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. 2018b. Semeval 2018 task 2: Multilingual emoji prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 24–33.
- Christos Baziotis, Nikos Athanasiou, Georgios Paraskevopoulos, Nikolaos Ellinas, Athanasia Kolovou, and Alexandros Potamianos. 2018. Ntua-slp at semeval-2018 task 2: predicting emojis using rnns with context-aware attention. *arXiv preprint arXiv:1804.06657*.
- Man Liu. 2018. Emonlp at semeval-2018 task 2: english emoji prediction with gradient boosting regression tree method and bidirectional lstm. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 390–394.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. 2019. Federated learning for emoji prediction in a mobile keyboard. *arXiv preprint arXiv:1906.04329*.

# Why Is Emoji Prediction Difficult?

Matija Bertović, Antun Magdić, Ante Žužul

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia  
{matija.bertovic, antun.magdic, ante.zuzul2}@fer.hr

## Abstract

With the rise in popularity of social networks such as Facebook, Twitter and Instagram, emojis have become more prominent and ubiquitous than most people could have expected. Nowadays, we use them for various purposes. Since they carry additional information, they could be of great use for various NLP tasks. The same way word prediction helps models to "understand" words, emoji prediction could help them "understand" emojis. But the task of emoji prediction is much harder than the task of predicting words. In this paper we compare simple models on the task of emoji prediction with the goal of identifying the main difficulties. We then cluster emojis to show main sources of problems occurring in their prediction.

## 1. Introduction

It is not wrong to say that the world without emojis is unimaginable. Those miniature pictures have an enormous impact on our lives, probably much more than we actually realize. Billions of emojis are used by people all around the world every day to better express themselves. Some use them to add emotion to the text, some to better communicate the message they want to send, some to indicate sarcasm to blur the line between written and in-person communication even more and some just because they like them. Sometimes emojis are used to add subtle details to the text, while at other times they are used to completely replace the text. And they often replace it very well. One emoji can often contain as much, if not more, information as multiple words.

It is obvious that since emojis contain a lot of information, their understanding could improve performance on many NLP tasks. So far, the majority of work handled the emoji prediction in a similar way and produced satisfying results. However, this problem might be too difficult the way it is currently approached.

The task seems simple to most people because almost everyone today has an autocomplete keyboard that suggests emojis while the person is typing the text. It seems that suggestions are solid: every time one types the word *sad*, 😞 is suggested. But it is not very often that people really choose the suggested emoji. People (still) have their own minds and decide by themselves which emoji they want to use. There are many factors that impact the choice of emoji. If the model is to be successful at the task of emoji prediction it should understand the majority of those factors. Often, not all of those factors are included in the text which contains emojis.

In this paper we run two experiments. First, we compare some simple models at the task of emoji prediction to gain some insight about the problem. After that, we cluster emojis and point to some issues that should be resolved before emoji prediction could substantially improve.

One great thing about emoji prediction is that there is much available and easily obtainable data. There are billions of new emojis occurrences on Facebook, Twitter, Instagram and various other social networks every day. In

this paper we use data from Twitter. Ten million tweets are collected and tweets with most frequent emojis are used for the task of emoji prediction.

## 2. Related Work

Over time, emojis evolved into ubiquitous and powerful communication mechanism, especially on social networks. Therefore, there has been a rise in the number of studies trying to understand them better and even predict them. As far as we know, most of the previous works are mainly focused on emoji prediction, rather than examination of their true meaning.

State-of-the-art results for emoji prediction task are obtained mostly by models based on bidirectional LSTMs (BLSTMs) (Barbieri et al., 2017a). In addition to BLSTM, attention mechanism ensures dropout of less informative words (Felbo et al., 2017). Another interesting approach to the emoji prediction problem is an investigation of the temporal information impact on emojis (Barbieri et al., 2018a). It is shown that the usage of some emojis varies depending on the time of the year.

On the other hand, emoji prediction seems impossible if the meaning of emoji is unknown. Barbieri et al. (2016) used skip-gram embedding model for quantitative and qualitative evaluation of Twitter emojis. Vector pair similarity and relatedness tests were performed, as well as clustering. Each emoji was represented with cluster containing most similar text tokens.

## 3. Dataset

We frame the task of emoji prediction as a supervised learning task. Each example is made of a tweet labelled with the emoji it contains, which is removed from the tweet body as in (Barbieri et al., 2017b).

We gathered ten million tweets from the period between November 1, 2018 and December 31, 2018. From those tweets we extracted only the ones which contain a single emoji. In the final dataset we kept only the tweets where one of the 20 most frequent emojis occurs. We split the data in train, validation and test sets containing 120 000, 40 000 and 40 000 tweets, respectively. Classes in all sets

are perfectly balanced<sup>1</sup>. Unfortunately, the dataset contains a portion of *re*-tweets, meaning that some tweets’ contents in the dataset are present multiple times across train, validation and test set.

## 4. Models and Representations

We experimented with various models. Different models use different input representations which include binary bag of words vectors, TF-IDF vectors (Manning et al., 2008), as well as 100 dimensional GloVe word embeddings pretrained on Twitter data (Pennington et al., 2014).

In the following subsections  $\hat{y}$  is used to denote the predicted class, and  $\mathcal{Y}$  is used to denote the set of all classes. The classes are labelled by integers ranging from 1 to 20, so  $\mathcal{Y} = \{1, 2, 3, \dots, 20\}$ .

### 4.1. Naïve Bayes

Naïve Bayes (Manning et al., 2008) is a probabilistic model for classification. It takes advantage of the Bayes rule to compute the probability

$$P(y|\mathbf{x}) = \frac{\mathcal{L}(y|\mathbf{x})P(y)}{P(\mathbf{x})},$$

where  $y$  is the class label,  $\mathbf{x}$  is the example to be classified and  $\mathcal{L}$  is the likelihood function. Example is then assigned to the class  $\hat{y}$  with the highest probability:

$$\hat{y} = \operatorname{argmax}_{y \in \mathcal{Y}} P(y|\mathbf{x}).$$

When using this model, we represent each tweet with a binary bag of words vector and we use multivariate Bernoulli distribution as the likelihood function, where we make the naïve assumption of conditional independence of words in a tweet, given the tweet’s class label.

### 4.2. Logistic Regression

Logistic regression (Murphy, 2012) is a simple discriminative model. We train a logistic regression classifier for each class. The output of the classifier trained for the class  $y$  is the predicted probability that the given example belongs to the class  $y$ . The probability is given by

$$P(y|\mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x} + b)}},$$

where  $\mathbf{w}$  and  $b$  are learned parameters. We then use OVR strategy (Bishop, 2006) to make the final classification. Class with the maximum predicted probability is assigned to the input example, that is

$$\hat{y} = \operatorname{argmax}_{y \in \mathcal{Y}} P(y|\mathbf{x}).$$

We use this model with two different input representations: TF-IDF vectors and mean vectors of GloVe word embeddings of all the words in the tweet. In both cases we set the regularization parameter to 0.1.

### 4.3. Feed Forward Neural Network

Neural networks have shown to be strong performers at solving various problems, so we also use them for the task of emoji prediction.

We train two feed forward neural networks. One uses TF-IDF vector of a tweet as the input representation, while the other uses the mean vector of GloVe word embeddings of all the words in the tweet.

We use one hidden layer with size 100 in the network with TF-IDF input representation and we use three hidden layers with sizes 150, 100, 50 in the network with mean GloVe input representation. We set the regularization parameter to  $10^{-5}$  for both networks.

### 4.4. Bidirectional LSTM

A class of neural networks that performs remarkably well on NLP tasks are recurrent neural networks. Hence, we also use a Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997).

LSTM is a type of recurrent neural network that is able to capture long-term dependencies. Fully-connected layer is added after the LSTM cell to map the output of the LSTM cell to the vector of class logits. The final output of the network, i.e. the predicted class  $\hat{y}$ , is the class with the highest logit value:

$$\hat{y} = \operatorname{argmax}_{y \in \mathcal{Y}} (\mathbf{W}\mathbf{o} + \mathbf{b})_y,$$

where  $\mathbf{o}$  is the output of the LSTM cell and  $\mathbf{W}$  and  $\mathbf{b}$  are learned parameters of the fully-connected layer.  $y$  is used to index the output vector of logits, so  $y$  for which the highest logit is obtained is selected as the predicted class.

Two bidirectional LSTM (BLSTM) layers with hidden state size of 300 are used in the LSTM cell. A single bidirectional LSTM layer is composed of two standard LSTM layers, where one is processing the input sequence from the first word to the last word and the other is going the opposite way. This way, both past and future context is available at every time step. Both of those layers’ outputs are then concatenated into a single output vector of size 600. After the first BLSTM layer, a dropout layer with dropout probability 0.2 is used. In the end, a fully-connected layer with output size of 20 is used, because there are 20 different classes.

Parameters are optimized using ADAM (Kingma and Ba, 2014) with the initial learning rate of  $10^{-3}$ . The model is trained for 20 epochs over the train set with the batch size of 32.

Each input tweet is represented by a sequence of GloVe word embeddings.

## 5. Results

We run two experiments. In the first experiment we compare various models and their performances on the task of emoji prediction and in the second one we try to gain some insight about the use of emojis in tweets and the difficulties in their prediction.

<sup>1</sup>... as all things should be.

Table 1: Accuracy of various models on test data. NB stand for Naïve Bayes, LR for logistic regression, NN for feed forward neural network and BLSTM for bidirectional LSTM.

Model	Accuracy (%)
NB	51.15
LR GloVe	33.78
LR TF-IDF	53.35
NN GloVe	45.67
NN TF-IDF	51.05
BLSTM	51.40

### 5.1. Experiment 1

In experiment one we compare the performance of models described in Section 4.. It is important to stress out that our goal here was not to create a model that will achieve state-of-the-art results, but to experiment with different models and representations, compare them and try to understand the results. We also wanted to identify the difficulties of achieving high accuracies on this task, which is more thoroughly done in Experiment 2.

Achieved accuracies of the models are presented in Table 1. Only accuracies are shown since classes in test set are balanced.

The best accuracy is obtained by logistic regression that uses TF-IDF representation.

We first compare the models without temporal information so BLSTM is left out for now and will be tackled later. It is clear from Table 1 that the models which use TF-IDF representations perform better than the ones that use mean GloVe representations. Feed forward neural network with TF-IDF representations achieved 51.05% accuracy and logistic regression with TF-IDF representations achieved 53.35% accuracy, while the same models with GloVe representations achieved 45.67% and 33.78% accuracy, respectively. This is to be expected since TF-IDF vectors contain much more information than mean GloVe vectors, which are basically tweets reduced to a single word (that ideally combines the senses of all the words in the tweet).

More interesting result is the following. Logistic regression with TF-IDF representations outperforms BLSTM by almost 2%. BLSTM is a very powerful model with a lot of hyperparameters, so it can be tuned to perform better than it did, but again it was not our goal to achieve the best possible accuracy<sup>2</sup>. Results still show that taking into account temporal information, i.e. the exact word order when predicting emojis may not bring a lot more crucial information to the model. What is more important is the general information of the words from the tweet.

Another interesting, and for us surprising, fact is strong performance by the Naïve Bayes model. We conclude that the data doesn't strongly violate the naïve assumption. It also signals that TF-IDF representation maybe doesn't of-

<sup>2</sup>We were also constrained by the available computing power.

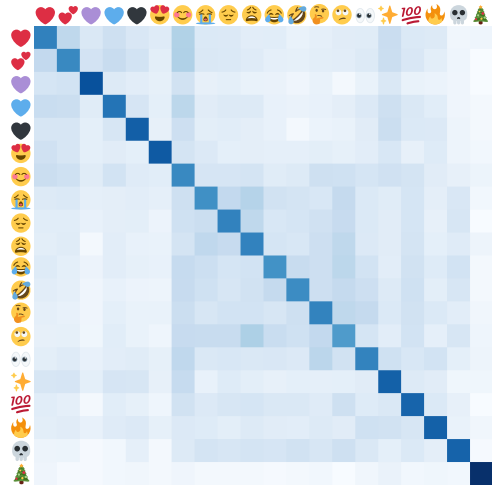


Figure 1: Confusion matrix for feed forward neural network with TF-IDF representations. Values shown are square roots of real counts (for better visibility).

fer much improvement with respect to binary bag of words representation.

In Figure 1 the confusion matrix for feed forward neural network using TF-IDF representation is shown. Some interesting information can be extracted from the confusion matrix. It can be seen that the model often confuses emojis in the following groups: ❤️, 🍀; 😞, 😓; 😞, 🤔; 😞, 🤔 and 😞, 🤔. This is evident from the squares along the main diagonal. Emojis in those groups convey the same meaning so it is reasonable that the model has trouble choosing between them. This issue will be explored in more detail in the following section.

### 5.2. Experiment 2

In the second experiment we cluster the tweets using  $K$ -means clustering (Bishop, 2006). The motivation behind this is as follows: if the quality of clustering is good with respect to class labels (emojis), i.e. tweets which include the same emoji are often found in the same cluster, the task of emoji prediction is probably not very hard. On the other hand, if the quality of clustering is poor, the prediction is probably hard since tweets with different emojis are not easily separable.

In this experiment we represent each tweet by the mean vector of GloVe word embeddings of all the words in the tweet. We believe this representation choice is justified by the results of the Experiment 1, where we show that temporal information is not crucial for emoji prediction. As can be seen from Table 1 the performance of neural network which uses mean GloVe vector as the input representation (model NN GloVe) achieves good enough accuracy compared to the models using TF-IDF vectors and sequences of GloVe word embeddings to justify using mean GloVe vectors as representations in our clustering experiment, especially since the goal here isn't to develop a state-of-the-art method, but only to deepen the understanding of the task of emoji prediction.

We run the experiment for various number of classes be-

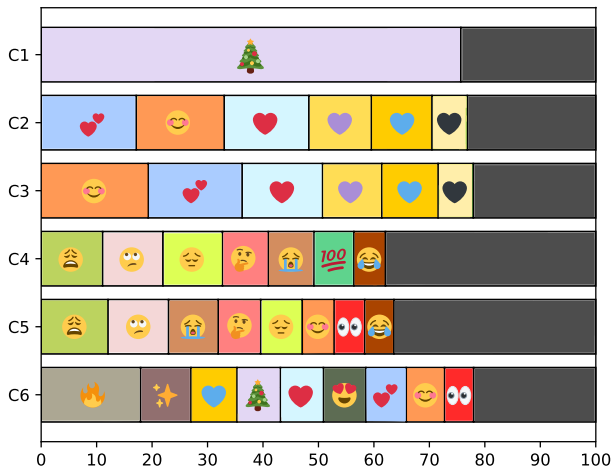


Figure 2: Some of the more interesting clusters. Each row represents a cluster. Percentages of the emojis contained in the cluster are shown on the x-axis. Only emojis that constitute more than 5% of the cluster are shown, while others are aggregated in the dark grey areas on the right side of each cluster.

tween 20 and 100. The results are qualitatively similar in all cases. Most clusters include a mix of emojis without the clear winner, so we present here a few interesting and informative examples. Clusters are shown in Figure 2.

Most of the tweets in cluster C1 are Christmas themed and contain 🎄. They are perfect examples of tweets whose class (containing emoji) is easy to predict. The theme of the tweets is very clearly expressed (very often by the exact word *Christmas*) so even the very basic models will have no trouble in classifying them, which can be seen easily in Figure 1. However, there are still more than 20% of tweets in this cluster that don’t contain 🎄. Most of them also have the similar Christmassy meaning, but their authors chose a different emoji to accompany the message<sup>3</sup>. There is no way, and it is unreasonable to expect, that any model might predict all of those emojis, for a model can only learn to assign a single emoji to a specific tweet content, but different users might choose different emojis. It is obvious that mere tweet content in many cases won’t be enough to make a good prediction.

Clusters C2 and C3 show the problem of synonymy among emojis. It is obvious that most of the tweets in those clusters convey a warm, loving, and in most cases, romantic message. In both clusters heart emojis make up more than 60% of emojis which makes it relatively easy for most models to recognize the general theme. But even in our limited set of 20 most frequent emojis, there are 5 heart emojis which act as synonyms. So even if the model is able to identify the general meaning of the tweet, it is still very hard for the model to predict the exact emoji. The reason is again that not all users will choose the same emoji to convey the meaning of romantic love, and without some background information about the author of the tweet, the model cannot make an informed decision.

<sup>3</sup>There are also quite a few *good morning* tweets in this cluster.

Clusters C4 and C5 convey mostly negative emotions: sadness, confusion, annoyance and grief. It is surprising to find 🙄 and 😞 in the mix. But the reason is pretty simple: those emojis are used here mostly in sarcastic setting. One could think that since there are many sarcasm detection systems available, that they would be able to solve this problem. Unfortunately, that might not be of much help because after removing the emoji most of those tweets don’t seem sarcastic anymore, for it is the emoji that signals the sarcasm. With emoji removed those tweets look like all the regular tweets with negative sentiment, and most models would assign them the emoji accordingly. This could be investigated further, especially with advances in sarcasm detection in mind.

Cluster C6 represents tweets with general positive sentiment that deliver joyful messages. In most cases it is hard to pin point the emoji. For example in the tweet

I just love when we all get together!! 🎄

it is very hard to predict the used emoji is 🎄. It might help if the model had some additional information about the tweet, like the time it was tweeted (which is December 25 in this example) and this was investigated in more detail in (Barbieri et al., 2018b). What could also help the model to predict the correct emoji is the picture that might accompany the tweet. If the example tweet included a picture of a family and a Christmas tree in the background (which is not unlikely) the problem would be much easier.

In other tweets in cluster C6 the problem of synonymy is apparent again. Some users might show joy and happiness using 😊, while others might use 🔥 or 🙄. Information about a user profile would without a doubt be helpful in emoji prediction.

## 6. Conclusion

In this paper we investigated the difficulties that arise during the task of emoji prediction. Even though the task seems simple on the surface, there are many subtle problems that need to be solved in order to create a well performing system for emoji prediction. We have shown that the crucial obstacle in achieving such a goal is synonymy among various emojis. To convey the same sense and meaning different tweet authors choose emojis in different ways. To achieve better performance at emoji prediction the model would have to be fed more information about the user. In that way the model could identify which emoji from the group of synonymous emojis the author would most likely pick (based on his previous tweets or other information). This is a pattern that emerges in various NLP and AI tasks: background knowledge can be helpful and offer crucial information for prediction.

We don’t expect that models for emoji prediction will improve substantially as long as information about the authors is not provided. As future work models that use author profiles for more informed prediction should be developed.

## References

Francesco Barbieri, Francesco Ronzano, and Horacio Sagion. 2016. What does this emoji mean? a vector space skip-gram model for twitter emojis. 05.

- Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017a. Are emojis predictable? *CoRR*, abs/1702.07285.
- Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017b. Are emojis predictable?
- Francesco Barbieri, Luís Marujo, Pradeep Karuturi, William Brendel, and Horacio Saggion. 2018a. Exploring emoji usage and prediction through a temporal variation lens. *CoRR*, abs/1805.00731.
- Francesco Barbieri, Luis Marujo, Pradeep Karuturi, William Brendel, and Horacio Saggion. 2018b. Exploring emoji usage and prediction through a temporal variation lens.
- Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.
- Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1615–1625, Copenhagen, Denmark, September. Association for Computational Linguistics.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80, 12.
- Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.
- Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge university press.
- Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. The MIT Press.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

# Depression Detection in Online Forums using Stylistic Features and Random Sampling

Vedran Bogdanović, Adi Čaušević, Juraj Vladika

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia  
{vedran.bogdanovic, adi.causevic, juraj.vladika}@fer.hr

## Abstract

Today, more than ever, mental health issues are a big problem for people of all ages and in all parts of the world. In a society dependent on the Internet, many try to find a refuge in social media where they communicate and express their thoughts. Their language use in combination with artificial intelligence models that are able to analyze immense amount of data, can help detect depression from clues that even humans might not be able to recognize. Our model, trained on a dataset extracted from the social media platform Reddit aims to do just that using a deep learning LSTM model combined with stylistic features and Monte Carlo random sampling. It achieves an F2 score of 0.7065 in detecting users with an early onset of depression.

## 1. Introduction

Internet has made the world a globally connected place and allowed individuals to discuss, express themselves and share their thoughts on various matters. All human communication is done through the use of language. Person's use of written language always carries implicit information about author and is a powerful indicator of personality, social or emotional status, but also their mental health. This is why a lot of work in the field of natural language processing (NLP) has been done in author profiling, more precisely predicting their age, gender, personality type, occupation, and other characteristics based solely on the language they use. In the same vein the link between language use and clinical disorders has been studied for decades. Social media have dominated the Internet discourse of 2010s and text written on them provides a means for capturing individual's behavioral attributes related to thinking, mood, communication and socialization. Postings may indicate feelings of worthlessness, guilt and helplessness, that all characterize mental disorders.

Major depressive disorder (MDD), or just depression, is a mental illness affecting more than 264 million people around the world according to World Health Organization.<sup>1</sup> It is characterized by long periods of bad mood, low self-esteem, loss of interest in enjoyable activities. It brings negative effects to personal and professional life, affecting the patients' education, sleeping and eating habits, and general health from heart to digestion. It can lead to disability, to psychotic episodes, and even to suicide. Up to 8% of adults with major depression die by suicide and about 50% of people who die by suicide had depression or another mood disorder.

In most cases depression stays undetected and untreated during a person's life. Current technology deployed to assess depression and suicide risk online is only reactive, i.e. action is taken only when an individual expresses intentions to harm themselves and at that point it can be too late to

help them. Therefore building a model that would help detect early onset of depression is a crucial task in helping those in need and consequently helping the general health of whole population.

In this paper we propose a deep-learning model enhanced with author's stylistic and general linguistic features, combined with a novel but powerful subsampling approach, which processes user's postings to detect the possibility of them having an onset of depressive disorder. Section 2 gives an overview of other authors' contributions to this task. The third chapter describes how the dataset for this task was collected and preprocessed. The fourth chapter describes the model architecture. The fifth chapter presents the stylistic features we used in the model. The sixth chapter shows results of our work and gives an analysis. The seventh and final chapter gives conclusion and future work.

## 2. Social Media and Depression

There is an undisputed connection between individual's language use and mental state. Early models were rule-based and incorporated linguistic rules observed by clinical professionals. The majority of them being the increased use of first-person pronouns as opposed to other pronouns, but also others such as usage of negation and words with negative valence, extensive past tense use for verb actions, inverted word order for topic, use of emphasis, presence of short, impersonal, truncated sentences, presence of ellipsis, tautologies, repetition, lack of comparison, and others (Trifu et al., 2017). Recently, machine-learning models have been deployed to make use of large quantities of data generated by users of Internet.

The most commonly used social network in this task was Twitter, with its large quantity of data making it convenient for research. Some successful approaches included using multiple channel convolutional neural networks (CNNs) (Orabi et al., 2018), graph theory for modeling social engagement with other users (Choudhury et al., 2013), multi-task learning predicting multiple mental disorders at once (Benton et al., 2017), interactive topic modeling beyond LDA (Resnik et al., 2015) and others. Twitter's character

<sup>1</sup><https://www.who.int/news-room/fact-sheets/detail/depression>

limit and unstructured nature of data can at times be limiting for more verbose thought expression. Social sites with longer posts have also been studied, e.g. Facebook where Schwartz et al. (2014) found a link between season of the year and time passage in degree of user’s depression, as well as specialized depression forums where Sadeque et al. (2016) examined why do users leave the forums after some time.

Reddit as a social platform with numerous specialized communities brings a lot of potential in studying this task because its anonymity provides users with the opportunity to be more open about their emotions and thoughts. There have been relatively few studies on this task using it. Some of them found reddit’s anonymity gathers feedback that is more involving and emotionally engaging (Choudhury and De, 2014), and that the diversity of topics users participate in showcase their language use in many different registers which helps in detection of self-harm (Yates et al., 2017). Where our paper differs from these is that it incorporates topic detection for placing posts in context, sentiment analysis to assess word valence, frequency metrics TF-IDF to single out most important depression-related words, and on the technical side a powerful approach of Monte Carlo subsampling, i.e. random sample selection of sequences in user’s post history – to the best of our knowledge, such an approach was not used for this task before.

### 3. Dataset

The dataset we used was collected and first described in a work by (Losada and Crestani, 2016). It is a collection of 892 reddit users’ post history. The dataset was split into 486 users (83 positive, 403 negative) in the train set and 406 (54 positive, 352 negative) in the test set. All users have at least 10 posts in post history and on average they had 597 submissions with average submission length of 33 words. Positive users were determined as such by looking for posts that contain explicit mention of user being diagnosed with depression by a professional. Later those posts were removed in order to remove bias. Negative users weren’t selected completely randomly, but were drawn from those users discussing depression and mental health in appropriate sections, often being friends and relatives of those who are diagnosed, and those who have post history in similar subreddits as positive users. There is no guarantee of users’ sincerity which means some noise in the data is to be expected. Due to training set’s small size the extraction of a validation set from training set was hard and gave improper results.

We preprocessed the dataset by removing punctuation and stop words (except for pronouns which were deemed important), concatenating user’s posting history into a single piece of text (using the space character as the separator), tokenizing it into tokens and stemming them using the Snowball Stemmer. For this the NLTK library<sup>2</sup> and Python 3.6 were used.

<sup>2</sup><https://www.nltk.org/>

## 4. Model Description

We used deep learning models which have been popular and widespread for NLP tasks in recent years and those are recurrent neural networks (RNNs).

### 4.1. Baseline

For the first baseline we used a simple recurrent neural network. The inputs to the model are pretrained 128 dimensional word2vec embeddings that represent the words in a dense form (Mikolov et al., 2013). The embeddings have been trained on the training set of our dataset and not the whole dataset, to prevent overfitting. The hidden layer of the RNN is 128 dimensions wide, while the output is a single value compressed to the range  $[0, 1]$  using a sigmoid function. The output represents the probability of the given text being written by a user with a depressive disorder.

### 4.2. LSTM

For our main model we used a single-layered long short-memory (LSTM) network. The input to the LSTM layer is consisted of pre-trained word embedding, concatenated with the values of features described in the next chapter. The output of the LSTM layer is forwarded to a fully connected layer with only one output. In the end that output is mapped to the range  $[0, 1]$  using a sigmoid function.

For training we used the ADAM optimizer with a learning rate of 0.002 which was adjusted using an exponential learning rate optimizer every 1000 iterations with the gamma factor set to 0.95. To prevent overfitting we also used L2 regularization where the weight decay was set to 0.001.

### 4.3. Monte Carlo Subsampling

Monte Carlo methods are, in general, a class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle. We decided that input to our model are going to be  $N$  randomly selected sequences of a fixed maximum length  $M$ . We opted for this approach because we noticed user’s whole post history was often too large and unrepresentative of his mental state, while single posts varied too much in length to single them out. This way we tackled both the problems with a highly successful approach recently used to beat the game of Go and other interesting problems (Silver et al., 2017).

For each of these input sequences model gives a probability of them being positive for depression. The final verdict on user’s depression status was determined by comparing the final summed score with a threshold. The basic algorithm for the MC subsampling method is displayed as follows:

```
function MC_SUBSAMPLING(user, num_samples,  $\phi$ )
    score  $\leftarrow$  0
    for  $i \in \text{range}(\text{num\_samples})$  do
        sample  $\leftarrow$  rand_sample(user.text)
        input  $\leftarrow$  embeddings(sample)
        output = model(input)
        score += soft_step(output)
```

**end for**

**return**  $score > \phi$

**end function**

Hyperparameters of this algorithm are the number of samples tested and the threshold for the score above which the user is classified to have depression. Within the algorithm soft-step function is used to adjust output of the model. Using this function the polarity of the prediction is exaggerated, meaning values above 0.5 will be pushed towards 1, while smaller values are pushed towards zero. Formula is defined as

$$softstep(x) = \frac{1}{1 + e^{\delta \cdot (0.5 - x)}}, \quad (1)$$

where  $\delta$  represents the slope of the shifted sigmoid function.

We found that by raising the number of samples the total running time of the algorithm increased significantly. Another thing to consider is that after a certain amount of samples tested, testing additional subsequences does not yield any new information as it is likely that most of the post history has already been tested. Having considered all that, we determined that a good number of tests is 20, while the threshold depends on the model used and mostly falls in range [1.5, 3].

## 5. Features

In this work, we defined four features as input to deep learning models. These features are used to emphasize key difference between depression positive and negative users. All of the features are dynamic, in sense that they are calculated on a sequence of fixed maximum length, as described earlier. In this way model is generalized because features do not depend on large amounts of data, but can be used with shorter excerpts (subsequences of text).

### 5.1. Pronouns ratio

Several theories suggest that self-focused attention plays an important role in the maintenance of depression (Zimmermann et al., 2016). To implement this phenomena in our model, we decided to measure the use of first-person and third-person pronouns. We argue that the ratio between occurrence of these pronouns can be an indicator of depression. In a short analysis of training dataset we found that ratio of total occurrences of first-person pronouns to third-person pronouns in positive users was 5 : 1 while in negative users 3.5 : 1 which already gives light to this hypothesis.

### 5.2. Sentiment

Sentiment analysis focuses on extracting emotion from written text. It can be used for detecting depression symptoms, like anger outbursts and sadness. We used a learned model from *textblob*<sup>3</sup> library to measure sentiment. This model returns a value for sentiment between -1 and 1, where -1 represents the most negative sentiment and 1 the most positive sentiment. Instead of averaging the sentiment of whole post history, we only measure sentiment on a sequence of fixed maximum length.

<sup>3</sup><https://textblob.readthedocs.io/en/dev/>

### 5.3. TF-IDF

TF-IDF is the weight computed as product of term frequency an inverse document frequency. Term frequency and inverse document frequency are calculated with formulas 2 and 4 respectively. The  $\mu$  factor, calculated using the formula 3, normalizes the *idf*-score for smaller values. The formula and its parameters were constructed so that  $\mu$  dampens the TF-IDF values words with a small document frequency.

$$tf(t_i, d_j) = 0.5 + \frac{0.5 \cdot freq(t_i, d_j)}{\max_{t \in d_j} freq(t, d_j)} \quad (2)$$

$$\mu(t_i, D) = \tanh\left(\left(\frac{1}{40} \cdot |\{d_j \in D | t_i \in d_j\}|\right)^3\right) \quad (3)$$

$$idf(t_i, D) = \log \frac{|D|}{|\{d_j \in D | t_i \in d_j\}|} \cdot \mu(t_i, D) \quad (4)$$

In all the formulas  $t_i$  is the word (term),  $d_j$  document (user's post history) and  $D$  document set (user set). The motivation behind this was to find those words that carry the biggest informational value because their appearance in user's posts implicitly tells about his language use.

### 5.4. Topic Modeling

We assumed that model has to have a good context-awareness in order to predict depression, as shown in (Gong and Poellabauer, 2018). This is why we introduce topic modeling as our final feature. Latent Dirichlet Allocation (LDA) algorithm from *gensim*<sup>4</sup> package is used to extract 12 topics from training data posts. The number of topics was chosen considering the number of user posts. It constructs the topics considering the most common and important words as defining a topic. The noun-only approach is used, as Martin and Johnson (2015) suggested. Feature is then calculated as the dominant topic in the current subsequence.

## 6. Results

### 6.1. Metrics

There are several commonly used metrics for classification problems, one of them being the F1 score. The F1 score is an accuracy metric which considers both the precision  $p$  and the recall  $r$  to compute an overall accuracy score, finding their harmonic mean with formula below

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \quad (5)$$

When using the F1 score we make the assumption that both precision and recall are equally as important. That might hold in most cases, but because of the nature of the problem, we found recall to be more important. As in most medical classification problems, labeling a healthy person as being depressive is less of an issue than mislabeling depressive person as being healthy. Considering that, we used  $F_\beta$  score defined as

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}. \quad (6)$$

<sup>4</sup><https://radimrehurek.com/gensim/>

Table 1: Model results on test set

Model	feature	window size	threshold	w2v-dim	precision	recall	$F_2$
Baseline	-	500	2.0	128	0.3991	0.6315	0.5654
LSTM (1 layer)	TF-IDF	200	2.0	200	0.4138	<b>0.8481</b>	0.7009
LSTM (2 layers)	pronouns ratio	500	1.5	200	0.4146	0.8314	0.6920
LSTM (1 layers)	sentiment	100	3.0	200	0.4889	0.8148	<b>0.7065</b>
LSTM (1 layers)	topic modeling	500	3.0	200	<b>0.4943</b>	0.7653	0.6897

Beta is a positive real number that tells how much more important is recall than precision. As we previously mentioned, the F1 score did not emphasize recall enough. On the other hand setting  $\beta > 3$  emphasized it too much, because of that we used the  $F_2$  score which we found to be a good compromise.

## 6.2. Analysis

Table 2: Average sentiment score in positive and negative examples

examples	sentiment
Negative	0.0919
Positive	0.1014

In Table 1 we compare the best results for every feature we introduced earlier. As we can see from the third and fourth column of the table, different models performed better with different parameters. Precision and recall were mainly influenced by threshold parameter. We can see that the models with smaller threshold values tend to have greater recall and lower precision. The overall winner is a model with sentiment feature. This may be because the sentiment of posts reflects user’s inner state and possible depression symptoms.

Another interesting thing that we found is that positive examples had a higher average sentiment score. This can be seen in Table 2.

As shown in Figure 1,  $F_2$  score varies greatly for different window sizes. Expanding the window size doesn’t necessarily improve performance, and while most models performed best for largest window size, model using sentiment modeling performed best for window size 100, and model using TF-IDF performed best for window size 200.

In the end we compared performances to show the improvement of the model when combined with Monte Carlo Subsampling. These results are shown in Table 3. Monte Carlo Subsampling method greatly improves Recall of the model, which is preferably as described in Metrics section.

## 7. Conclusion

In this paper, we presented an LSTM model equipped with stylistic features and Monte Carlo random sampling, that significantly improves depression detection of the baseline model. We confirmed that pronoun usage and putting posts

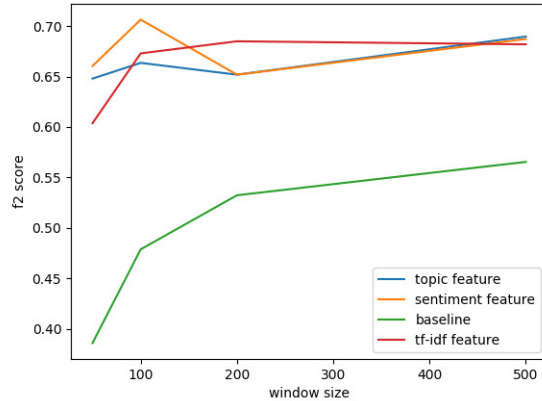


Figure 1: F2 scores for different window sizes

Table 3: Model performance with and without Monte Carlo Subsampling

Method	Precision	Recall	$F_2$
LSTM (without MCS)	0.3994	0.3352	0.3460
LSTM (with MCS)	0.3955	0.8315	0.6811

into context of topics help with determining depression presence. While exploring this new approach, we have uncovered multiple important insights, mainly that (1) sampling parameters are not consistent across different architectures; (2) contrary to beliefs, depressed patients don’t score lower in sentiment analysis, (3) contrary to previously popular use of F1 score for accuracy, we argued that  $F_\beta$  score, with beta higher than one, is more accurate score for depression detection and similar tasks. In our study, we demonstrated that the performance of existing RNN architectures can be consistently boosted.

For future work, we will evaluate random sampling approach on more architectures. We will also address the problem of inconsistent results of same models mentioned. We will also investigate model performance on other kinds of mental illnesses.

## References

- Adrian Benton, Margaret Mitchell, and Dirk Hovy. 2017. Multi-task learning for mental health using social media text. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 152–162.
- Munmun De Choudhury and Sushovan De. 2014. Mental health discourse on reddit: Self-disclosure, social support, and anonymity. *Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media*.
- Munmun De Choudhury, Michael Gamon, Scott Counts, and Eric Horvitz. 2013. Predicting depression via social media. *Proceedings of the Seventh International AAAI Conference on Weblogs and Social Media*.
- Yuan Gong and Christian Poellabauer. 2018. Topic modeling based multi-modal depression detection. *Proceedings of the 7th Audio/Visual Emotion Challenge and Workshop (AVEC)*.
- David E. Losada and Fabio Crestani. 2016. A test collection for research on depression and language use. *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 7th International Conference of the CLEF Association*, pages 28–39.
- Fiona Martin and Mark Johnson. 2015. More efficient topic modelling through a noun only approach. *Proceedings of Australasian Language Technology Association Workshop*, pages 111–115.
- T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. *In Advances in neural information processing systems*, pages 3111–3119.
- Ahmed Hussein Orabi, Prasadith Buddhitha, Mahmoud Hussein Orabi, and Diana Inkpen. 2018. Deep learning for depression detection of twitter users. *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, pages 88–97.
- Philip Resnik, William Armstrong, Leonardo Claudino, Thang Nguyen, Viet-An Nguyen, and Jordan Boyd-Graber. 2015. Beyond lda: Exploring supervised topic modeling for depression-related language in twitter. *Proceedings of the 2nd Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pages 99–107.
- Farig Sadeque, Ted Pedersen, Thamar Solorio, Prasha Shrestha, Nicolas Rey-Villamizar, and Steven Bethard. 2016. Why do they leave: Modeling participation in online depression forums. *Proceedings of The Fourth International Workshop on Natural Language Processing for Social Media*, pages 14–19.
- H. Andrew Schwartz, Johannes Eichstaedt, Margaret L. Kern, Gregory Park, Maarten Sap, David Stillwell, Michal Kosinski, and Lyle Ungar. 2014. Towards assessing changes in degree of depression through facebook. *Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, pages 118–125.
- David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017. Mastering the game of go without human knowledge. *Nature*.
- R. N. Trifu, B. Nemeş, C. Bodea-Hategan, and D. Cozman. 2017. Linguistic indicators of language in major depressive disorder (mdd). an evidence based research. *Journal of Evidence-Based Psychotherapies*, 17(1):105–128.
- Andrew Yates, Arman Cohan, and Nazli Goharian. 2017. Depression and self-harm risk assessment in online forums.
- Johannes Zimmermann, Timo Brockmeyer, Matthias Hunn, Henning Schauenburg, and Markus Wolf. 2016. First-person pronoun use in spoken language as a predictor of future depressive symptoms: Preliminary evidence from a clinical sample of depressed patients. *Clinical Psychology and Psychotherapy*.

# COVID-19 Information, Where Are You Hiding?

Marko Cvitković, Nikolina Čović, Patrik Marić

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia

marko.cvitkovic7@gmail.com, {nikolina.covic, patrik.maric}@fer.hr

## Abstract

Data on COVID-19 is currently of utmost importance and interest worldwide and to the medical research community in particular. With significant daily increases in the amount of data, potentially valuable information can easily be lost. To help prevent that, we present a retrieval system that gives the best fitting paragraph to a user query. We explore how different paragraph representations affect the performance of the engine and present the results on the COVID-19 data set, which is a collection of scientific papers on the subject of this new infection. Results show the precision of each representation on 8 COVID-19-related queries, with Word2Vec showing the best results.

## 1. Introduction

The first half of 2020 was marked by the spread of the novel coronavirus, named COVID-19, around the world. The virus was first detected in the Chinese province of Wuhan in early December, but until mid-February, when the virus penetrated Europe, the public interest was very low. After shocking footage from Italian hospitals went viral, the awareness of the danger of COVID-19 increased.

Out of human curiosity, but also out of caution to prevent the rapid spread of the virus, people wanted to inform themselves about the main characteristics of the virus. There are many sources regarding the topic of COVID-19 that can provide useful information, and the volume of it increases each day. A large amount of data is what most often results in the loss of potentially important information. Due to this rapid growth, we wanted to make the process of searching for relevant information more efficient and speed it up by extracting the parts of paper that carry the most relevant information to the user. Not only would that help the common man to obtain as accurate and as high-quality information about the disease as possible, but it could potentially benefit the medical research community with the fight against the novel coronavirus.

In this paper, a search engine that retrieves the most similar paragraphs from a corpus to a given query, based on cosine similarity, is presented. We wanted to examine how different paragraph representations, count-based and distributed, affect the performance of the search engine. The performance was evaluated on COVID-19 corpus, consisting of more than 100,000 scientific papers about COVID-19 and similar infections. Results show the precision for queries about COVID-19 main information and risk factors for different paragraph representations, with the average of Word2Vec representations performing as the best one.

The next section gives an overview of work on paragraph vector modelling and paragraph retrieval so far. Section 3. gives the background knowledge about the models used for generating different paragraph representations. In section 4., we give information about the data set, how the pre-processing and data annotation was done. The results are presented in section 5.

## 2. Related Work

The paragraph retrieval task depends heavily on the informative vector representations of the paragraphs. There have been many attempts to build the most informative and useful paragraph vector representations.

After presenting word representations in (Mikolov et al., 2013a), as weights of a hidden layer of the trained neural network, the research community turned to model more complex structures. (Wu et al., 2018; Mikolov et al., 2013b) try to model more accurate representations of word phrases and other word compositions.

A way to obtain paragraph vector representations is to use the topic modelling approach. One of the more popular methods of representing words and documents in latent topic space is Latent Dirichlet Allocation (LDA), first presented in (Blei et al., 2003). In our work, we did not try any topic modelling algorithm, but it would be interesting to see how it competes with models that we used.

Another way of getting paragraph vectors would be the encoder-decoder models, like Paragraph Vector introduced in (Le and Mikolov, 2014). This model benefits significantly from having a large amount of unlabeled data. It uses unlabeled data to learn feature representations of fixed size from variable-length inputs, such as sentences, paragraphs, documents.

The Transformer model (Vaswani et al., 2017) inspired others to use Transformers for incorporating sequence information in sequence representation. BERT, a very popular model presented in (Devlin et al., 2019) is just one of them. BERT produces high quality contextual token embeddings, which are of great use in a number of downstream NLP tasks, but not so much for getting the out-of-box paragraph embeddings as some pooling combination of BERT's output layers. There are highly efficient systems like SBERT (Reimers and Gurevych, 2019) or *bert-as-service*<sup>1</sup> that use the fine-tuned BERT models to get better sequence representations.

Paragraphs vector representations are used for different information retrieval tasks, such as QA systems presented

<sup>1</sup><https://github.com/hanxiao/bert-as-service>

in (Hoque and Quresma, 2015; Stone et al., 2011; Feldman and El-Yaniv, 2019; Salton et al., 1993). In our work, we focused on Google-like queries, but it would be interesting to see in future work how our representations perform on queries formulated as questions.

### 3. Background

We aimed to get a vector representation of every paragraph, and a compatible vector representation of incoming queries, a task for which we utilised six different models. Here, we briefly describe the details of each one.

#### 3.1. TF-IDF

The first version of our search engine implements a bag-of-words (BOW) approach. We feed preprocessed corpus in the form of a set of paragraphs to the model. First, we move from textual to a numerical type of data. Every paragraph is now a V-dimensional vector, where V stands for the dimension of the corpus vocabulary. All vector components correspond to the number of appearances for each vocabulary token. Then, the global IDF vector is learnt and a sparse count matrix is transformed into a TF-IDF representation. TF-IDF gives higher weights to words which appear more in a document, but at the same time appear less across other documents. All incoming queries are transformed based on corpus vocabulary, also yielding a V-dimensional vector representation. We ignore any potential out-of-vocabulary (OOV) words in a query.

#### 3.2. BM25

Here, the same frame is applied, as described in the previous subsection. The only difference came in the transformer component. Instead of transforming the count matrix into a TF-IDF one, we transformed it using the BM25 ranking function. BM25 is an empirically obtained expression, a function of TF-IDF and a few other chosen parameters which adjusts for previous similar models deficiencies concerning treating documents with different length. Again, we get the V-dimensional vector for each paragraph in the corpus and any given query.

#### 3.3. Word2Vec

The previous two approaches did not utilise syntactical nor semantic information encapsulated in the corpus, such as ordering between tokens, the meaning of words or context. Here, we train the Word2Vec (CBOW) model using the Gensim library, and we obtain a 300-dimensional dense vector representation for each word in vocabulary corpus. The hyper-parameters, given in Table 1, used for training the model, were chosen experimentally. We tried two different pooling strategies for combining word representations into paragraph vector.

The first method was just to average out the word embeddings:

$$paragraph\_vec(P) = \frac{1}{N} \sum_{w \in P} word\_vec(w)$$

where N represents number of words in a paragraph. In the second method, we tried the weighted average where

we use TF-IDF to calculate weights:

$$paragraph\_vec(P) = \frac{1}{N} \sum_{w \in P} word\_vec(w) \cdot tf\_idf(w)$$

To represent the queries, we use the same methods. We evaluated each of these two versions and present the results in section 5.

#### 3.4. Doc2Vec

To move one step further, we decided to generate vector representations directly for paragraphs without building them up from smaller units, and we picked Gensim’s Doc2Vec model to help us with that agenda. We used PV-DBOW method, which is essentially an adaptation of the Skip-Gram version of Word2Vec, to obtain 100-dimensional paragraph and query representations. Since the data set consists of 118,266 (53.3%) low-length paragraphs (less than 100-word tokens), lower vector dimension was chosen to capture the paragraph information better. Therefore we preliminary trained the model with 50, 100 and 300-dimensional vectors, out of which, 100-dimensional vectors provided the best-retrieved paragraphs. Additionally, the optimal number of training epochs in most of the papers containing Doc2Vec models ranges from 10 to 20. Since the improvement when training with 20 epochs wasn’t noticeable and the run time doubled in preliminary training, we decided to use 10 epochs. Other hyper-parameters were chosen experimentally and based on other papers, such as (Lau and Baldwin, 2016), and can be found in Table 1.

Table 1: Hyper-parameters used for training<sup>2</sup>

	Method	Min Count	Vector Size	Window Size	Sub-sampling	Negative	Epochs	Alpha	Min Alpha
W2V	CBOW	5	300	5	-	5	-	-	-
D2V	PV-DBOW	5	100	5	0	5	10	0.025	0.0025

#### 3.5. SBERT

(Reimers and Gurevych, 2019) show that average of BERT token-level output embeddings maps sentences to a vector space which is rather unsuitable to use with common similarity measures like cosine-similarity along with process being computationally infeasible.

So our last approach was to represent paragraphs via sentence embeddings generated with UKPLab’s<sup>3</sup> SRoBERTa-NLI-STSB-base model, fine-tuned specifically for semantic text similarity task, which encodes every input sentence to a 768-dimensional vector. Those embeddings showed significant improvement in encapsulating sentence semantics.

Here, we had three strategies for calculating the paragraph vector. The first choice was to average out all the sentence embeddings belonging to a given paragraph. The second approach was first to calculate similarities between a query and all the sentences in a paragraph and then assume that the paragraph is as valuable as is its most valuable sentence. The last strategy analysed was assigning a paragraph-query similarity as the average similarity between the query and

<sup>2</sup>Default values were used for hyper-parameters with no value

<sup>3</sup><https://github.com/UKPLab/sentence-transformers>

paragraph’s two most similar sentences. The motivation behind this was the fact that our shortest paragraphs had two sentences so we could not consistently explore more than the top two. We wanted to rely on more than just one sentence per paragraph because the sentences in the dataset are sometimes unfinished, without meaning or misspelt.

We analysed all three options and decided to evaluate and present the results in section 5. only for the second option because it showed the best performance. Queries are processed simply by getting the sentence embedding. It is worth noting that we did not evaluate this model on preprocessed data because preprocessing worsened the performance.

## 4. Experimental Setup

We compared the results of 6 different model approaches, described in section 3. After building our engine for each approach by obtaining and normalising vectors that represent all the paragraphs in the data set, we ran each query by calculating cosine similarities between every paragraph vector and generated query vector, as shown in Figure 1. Paragraphs are then sorted in the decreasing similarity order, and top 10 paragraphs for each query are taken into account for further analysis and annotation.

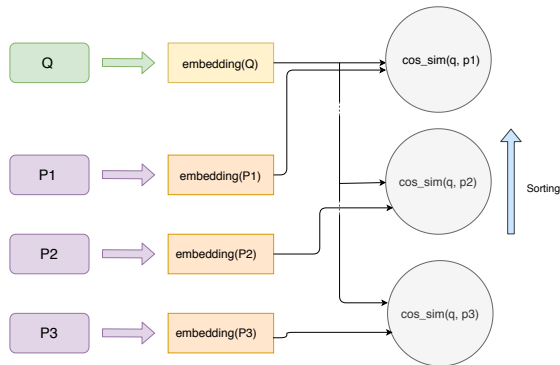


Figure 1: Architecture of the model. Engine is built by generating emeddings of all paragraphs. Query is entered, processed and top k paragraphs based on cosine similarity are retrieved.

### 4.1. Data Set

We experimented on the CORD-19 data set<sup>4</sup>, which was prepared by various research scientist groups to gain new knowledge and help the fight against COVID-19. At the moment of downloading, it contained 103,314 articles. The paper consisted of text in abstract and text in the body of the paper. After preprocessing, described in detail in section 4.2., the data set that we worked with consisted of 221,998 paragraphs. The average number of word tokens per paragraph was 118, while the minimum and maximum were 5 and 5952, respectively.

### 4.2. Data Preprocessing

Before use, the data set needed preprocessing in order to obtain better paragraph representations. Firstly, all the pa-

pers not containing key words<sup>5</sup> were removed, as well as all non-English papers. Additionally, we removed all paragraph duplicates and tokenized the paragraphs into sentences. To avoid unuseful information, we discarded those containing only one sentence, such as who wrote the paper or the year of publishing.

Then we tokenized the sentences of remaining paragraphs into words and converted them to lower case. If the sentence consisted of less than five words, we removed it from the paragraph due to the lack of potential for containing relevant information. We removed all punctuation and stop words, converted all numbers into words, and finally stemmed the words with Porter stemmer.

### 4.3. Data Annotation

Each model was ran on eight queries and the ten most relevant retrieved paragraphs were added to a pool. That left us with a total of 480 query-paragraph pairs, of which 389 were unique. After that, three annotators by majority rule voting decided whether the retrieved paragraphs were relevant to the query. We used this pool to evaluate the performance of the models and show the results in the next section. Inter-annotator agreement measure Cohen’s Kappa was 0.75, which is interpreted as substantial.

## 5. Results

The queries that we chose seek information about COVID-19 risk factors and basic information about the disease, such as the duration of the incubation period. To compare our models, we used the annotated retrieved paragraphs pool. For each model and each query, we calculated the ratio of retrieved paragraphs that was said to be relevant by the annotators. We show the results in Table 2.

Table 2: Model precision for each query, average model precision@10 with STD and MAP

Query	tf-idf	bm25	w2v	w2v_tf-idf	d2v	SBERT
Basic reproductive number of covid19	0.9	0.9	0.9	<b>1.0</b>	0.4	0.8
Comorbidities and coinfections with effect on the severity of covid19	0.2	0.4	0.9	<b>1.0</b>	0.6	0.5
Covid19 effect on pregnancy	0.3	0.2	0.7	0.2	0.5	<b>0.9</b>
Incubation period of covid19	0.9	0.3	<b>1.0</b>	0.9	0.7	0.1
Mortality rate of covid19	0.5	0.3	<b>0.9</b>	<b>0.9</b>	0.8	0.4
Smoking effect on the severity of covid19	0.9	0.3	<b>1.0</b>	0.9	0.7	<b>1.0</b>
Temperature effect on covid19	0.3	0.0	0.9	<b>1.0</b>	0.6	0.4
Ways of covid19 transmission	0.4	0.1	<b>0.9</b>	0.7	0.3	0.4
<b>Model precision@10</b>	0.55±0.30	0.31±0.27	<b>0.90±0.09</b>	0.83±0.27	0.58±0.17	0.56±0.30
<b>MAP</b>	0.57	0.44	<b>0.98</b>	0.95	0.67	0.79

Surprisingly, BM25 performed worse than TF-IDF for both precision@10 and mean average precision (MAP) scores. That can to a degree be attributed to specific selection of queries. Generally, we observed that count-based

<sup>4</sup>It can be found on: <https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge>

<sup>5</sup>Chosen keywords were *novel coronavirus, covid, wuhan, sars-cov-2, coronavirus, severe acute respiratory syndrome, corona*

models are biased and greedy towards paragraphs that mention a certain token as many times as possible, failing to recognise user’s information need expressed in a query. In a query about the basic reproduction number of COVID-19, one of the retrieved paragraphs by TF-IDF model was irrelevant and had COVID-19 mentioned many times.

Doc2Vec was at least partially successful on every evaluated query, but we think that there is a discrepancy generated between representing a paragraph and short (4 to 8 words) queries. It managed to retrieve several topic-related paragraphs, but it was related to another infection, which we annotated as irrelevant.

SBERT performance varied from worst to the best across different evaluated queries. We believe that the fact holding it down was the queries are not formed as fully functional sentences. As a result, generated query embeddings are not robust enough to capture the information need. For example, for the incubation period related query, it retrieved many paragraphs with mentions of a certain time period.

The two Word2Vec variations seemed to find the best balance and robustness capturing the matching semantics for both the query and paragraphs and showed to be the right model for the task at hand. Still, Word2Vec with average pooling performed better on our queries as we can see the TF-IDF weighting version performed extremely bad on the query about effects of COVID-19 on pregnancy. One example among many is shown, from where it is obvious that the term frequency component in weights for the word effect became dominant. As a result, the system retrieved irrelevant paragraphs.

In Table 3, we show both the relevant and mentioned irrelevant retrieval results.

## 6. Conclusion

Amid a global crisis caused by COVID-19 pandemic, we decided to tackle data behind the problem and make our humble contribution by creating a search engine for COVID-related information.

We retrieved paragraphs relevant to a set of specified handpicked queries and presented the system developed for the task. We used six different models, annotated relevant retrievals, calculated precision at top k and compared the results. Generally, distributed representations yielded better results for the task, namely the two Word2Vec versions.

We release all our code publicly on Github<sup>6</sup> along with this paper. It can be further improved by evaluating the models on a larger set of queries, exploring the results on different types of queries and by using different methods of hyper-parameter optimisation, instead of a manual search.

Different and model-specific preprocessing could be used and evaluated by ablation study of preprocessing components. Also, domain-based word and sentence embeddings can be implemented and tested.

<sup>6</sup><https://github.com/patrikmaric/COVID19-search-engine>

Table 3: Retrieval examples

<b>Q:</b> Basic reproductive number of covid19
<b>P:</b> For the Covid-19 pandemic caused by the SARS-CoV-2 virus, a basic reproduction number $R_0 = 2.6$ is estimated, with an uncertainty range from 1.5 to 3.5 [5].
<b>Similarity, Model:</b> (0.92, w2v_tf-idf), (0.58, tf-idf), (0.41, bm25)
<b>Q:</b> Comorbidities and coinfections with effect on the severity of covid19
<b>P:</b> Metabolic and cardiovascular comorbidities like diabetes and hypertension aggravate the & severity of COVID-19. [2] Another comorbidity, MAFLD, also affects COVID-19 severity,...
<b>Similarity, Model:</b> (0.84, w2v_tf-idf), (0.83, w2v)
<b>Q:</b> Covid19 effect on pregnancy
<b>P:</b> Based on most recent epidemiologic data on COVID-19 and pregnancy, there is no evidence to suggest increased risk for mothers or fetuses.
<b>Similarity, Model:</b> (0.74, w2v_tf-idf), (0.67, w2v)
<b>Q:</b> Incubation period of covid19
<b>P:</b> Incubation time for COVID-19 between 2 and 10 days has been recorded by the WHO 27 . 2. The Chinese National Health Commission initially expected a 10 to 14 days incubation period.
<b>Similarity, Model:</b> (0.77, d2v)
<b>Q:</b> Mortality rate of covid19
<b>P:</b> Moreover, recent COVID-19 mortality data shows that the average mortality rates, especially for younger age groups, are significantly lower than the ones given above...the average mortality rate to be 0.9 %
<b>Similarity, Model:</b> (0.74, d2v), (0.92, w2v_tf-idf), (0.84, w2v)
<b>Q:</b> Smoking effect on the severity of covid19
<b>P:</b> According to the latest study on COVID-19, 12.6 % were current smokers (137/1085), and these patients are more severe than never-smokers as indicated by the severe rate (smoker: 21.2 %, 29/137 vs. never-smoker: 14.5 %, 134/927) [8]. These data suggested that cigarette smoking may affect the pathogenesis and progression of COVID-19.
<b>Similarity, Model:</b> (0.85, SBERT)
<b>Q:</b> Temperature effect on covid19
<b>P:</b> The purpose of this paper is to improve understanding of the effect of environmental temperature on the spread of COVID-19 and its exponential growth rate.
<b>Similarity, Model:</b> (0.68, w2v), (0.68, d2v)
<b>Q:</b> Ways of covid19 transmission
<b>P:</b> Throughout, we have mostly considered disease spreading through person-to-person direct contact. COVID-19 can also spread through other means...it is still the person-to-person direct transmissions that are of the primary concern
<b>Similarity, Model:</b> (0.78, w2v), (0.86, w2v_tf-idf)
<b>Q:</b> Basic reproductive number of covid19
<b>P:</b> The pandemic of COVID19 is ongoing. Till April 2 2020, COVID19 has caused more than 965 thousand infections and 49 thousand deaths world widely. Current evidences indicate COVID19 can be transmitted presymptomatically [1] [2] .
<b>Similarity, Model:</b> (0.41, tf-idf)
<b>Q:</b> Covid19 effect on pregnancy
<b>P:</b> The results showed a therapeutic effect on the symptoms of angina pectoris: of the 42 patients, there were 15 patients with a marked effect, 23 patients with an effect, and 4 patients without effect. The effective rate was 90.48 %. Regarding therapeutic effects on the electrocardiogram ST-T segment, of the 42 patients, 11 patients had a marked effect, 16 patients had an improving effect, and 15 patients had no effect; the effective rate was 64.29%.
<b>Similarity, Model:</b> (0.84, w2v_tf-idf)
<b>Q:</b> Comorbidities and coinfections with effect on the severity of covid19
<b>P:</b> MERS-CoV also has higher replication rates and shows broader cell tropism in the lower human respiratory tract than severe acute respiratory syndrome (13). These results suggest that a shorter incubation period could be related to a higher initial infective dose and consequently to faster or greater pathogen replication. This finding could lead to a more severe disease.
<b>Similarity, Model:</b> (0.68, d2v)
<b>Q:</b> Incubation period of covid19
<b>P:</b> Diffusion of COVID19. Number of infected from 17 March, 2020 to April 2020 (Ministero della Salute, 2020).
<b>Similarity, Model:</b> (0.84, SBERT)

## References

- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.
- Yair Feldman and Ran El-Yaniv. 2019. Multi-hop paragraph retrieval for open-domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2296–2309, Florence, Italy, July. Association for Computational Linguistics.
- M. M. Hoque and P. Quaresma. 2015. An effective approach for relevant paragraph retrieval in question answering systems. In *2015 18th International Conference on Computer and Information Technology (ICCIT)*, pages 44–49.
- Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 78–86, Berlin, Germany, August. Association for Computational Linguistics.
- Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1188–1196. JMLR.org.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.
- Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11.
- Gerard Salton, J. Allan, and Chris Buckley. 1993. Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '93*, page 49–58, New York, NY, USA. Association for Computing Machinery.
- Benjamin Stone, Simon Dennis, and Peter Kwantes. 2011. Comparing methods for single paragraph similarity analysis. *Topics in Cognitive Science*, 3:92 – 122, 01.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.
- Lingfei Wu, Ian En-Hsu Yen, Kun Xu, Fangli Xu, Avinash Balakrishnan, Pin-Yu Chen, Pradeep Ravikumar, and Michael J. Witbrock. 2018. Word mover’s embedding: From word2vec to document embedding. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun’ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4524–4534. Association for Computational Linguistics.

# How Offensive is Negative Sentiment? Using Offensive Language Detection Models for Sentiment Classification

Katarina Čavar, Sanja Deur, Dorotea Protrka

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia  
{katarina.cavar, sanja.deur, dorotea.protrka}@fer.hr

## Abstract

Removing offensive content on social media was until recently a task that had to be done manually. With the rise of machine learning and natural language processing techniques, automated offensive language identification has become an active area of research. This paper explores a wide range of architectures of both shallow machine learning and deep learning models to accomplish this task. Since an offensive statement can often be viewed as a statement with a negative sentiment, the connection between the tasks of offensive language detection and sentiment analysis is investigated. For this purpose, the success of our models, pretrained on offensive language detection task, is evaluated for the sentiment analysis task, and the transferability of models across these two domains is discussed.

## 1. Introduction

Social media platforms, such as Twitter and Facebook, are a popular way for people to express their feelings and opinions on different matters. The problem arises when those posts start containing hate speech towards another person or a specific group of people, as well as abusive and offensive language. In recent years, there have been various attempts and proposals on how to address this matter, such as paraphrasing the offensive part of a sentence whilst keeping the overall sentiment, as proposed by (dos Santos et al., 2018), or complete removal of some posts, depending on the degree of profanity.

First step in tackling offensive language across social media most certainly is its detection. Considering the enormous amount of data retrieved from social media platforms, desirable solution would include automatic identification of profane language, without or with small amount of manual filtering, as it is very time-consuming and may even cause post-traumatic stress disorder-like symptoms to human annotators (Zampieri et al., 2019a).

Recently, a lot of research has been done in this area, and various exceptional models have been developed. The aim of this paper is not on developing a state-of-the-art model, but rather we pursue the construction of different models and comparison of their performance. Our starting point is the OffensEval-2020 shared task which provides the Offensive Language Identification dataset (OLID) with over a 14,000 English tweets (Zampieri et al., 2019b). Furthermore, the second experiment that we have conducted concerns measuring the performance of pretrained models which detect offensive language on the binary Twitter Sentiment Analysis (TSA) problem, through which we attempt to infer the connection and similarity between the two mentioned tasks.

The remainder of this paper is organized as follows: Section 2. presents the overview of prior work done in the field of offensive language identification. Section 3. includes a brief description of datasets and pre-processing techniques used in this work. Section 4. thoroughly describes constructed models, divided into three main categories: (1)

baseline model, (2) shallow models and (3) deep models. In Section 5. results are provided and discussed. Section 6. gives concluding remarks and directions for future work.

## 2. Related Work

Over the past few years, the field of offensive language detection has arisen a lot of interest in scientific community. One of the latest work is the OffensEval-2020 shared task, the successor of SemEval-2019 task (Zampieri et al., 2019b) which has greatly influenced our work. For the purpose of offensive language identification, authors usually decide between either using shallow machine learning models, deep learning approach or an ensemble of more methods.

On the one hand, different sorts of machine learning approaches are examined in (Davidson et al., 2017), and the authors finally decide upon using logistic regression with L2 regularization. Furthermore, (Nobata et al., 2016) argue that shallow machine learning based methods outperform deep learning ones, and they manage to outperform a state-of-the-art deep learning approach.

On the other hand, in recent years, deep learning models are gaining popularity and are shown as more successful ones. One approach in detection of offensive language proposed in (Pitsilis et al., 2018) is using ensemble of Long Short Term Memory (LSTM), a powerful type of Recurrent Neural Network (RNN), classifiers. Second interesting approach is a sequentially combined BiLSTM-CNN neural network combined with transfer learning (Wiedemann et al., 2018). The data used in the experiments are more than 5,000 German tweets composed for GermEval-2018 shared task, described in detail in (Wiegand et al., 2018). Lastly, the best solution regarding sub-task A of OffensEval-2020 shared task, (Liu et al., 2019), experimented with linear models, LSTM, and pretrained BERT with finetuning on the OLID dataset.

The connection between offensive language detection and sentiment analysis is undeniable, due to the fact that offensive language, in most cases, has a negative sentiment. The main difference is the fact that profane language could be

used in tweets with positive sentiment to express emphasis (e.g. tweet "ON HALLOWEEN its about to get sooooo F\*CKING spooky"). There are mentions in (Zampieri et al., 2019b) of successfully adapting BiLSTM model for offense detection task from a pre-existing model for sentiment analysis. Our goal is to apply the same model we implemented for offensive language identification on sentiment analysis task and check the performance, and consequently the similarity of those two tasks. To the best knowledge of authors, this idea has not yet been examined by others.

### 3. Dataset

#### 3.1. OLID Dataset

Offensive Language Identification dataset (OLID) consists of 14,100 annotated tweets in English. Labels provided for this dataset are divided into three categories based on the sub-task:

- Sub-task A: Offensive language detection
- Sub-task B: Categorization of offensive language types
- Sub-task C: Offensive language target identification

Since our interest lies in offensive language identification, only the first group of labels is used. Tweets are partitioned into two groups: offensive (OFF) and not offensive (NOT). Distribution of tweets is shown in Table 1. For training purposes, we further shuffled and divided the train set into train and evaluation sets.

Table 1: Distribution of OLID dataset

Label	Training	Test	Total
OFF	4,400	240	4,640
NOT	8,840	620	9,460
<b>Total</b>	13,240	860	14,100

#### 3.2. MTSA Dataset

McGill Twitter Sentiment Analysis dataset (MTSA) (Kenyon-Dean et al., 2018) contains over 7,000 tweets across five different topic-domains, annotated by at least five annotators. Tweets are categorized according to following four labels: objective, positive, negative and complicated. Label objective is used in case tweet does not express sentiment, and if it does, sentiment is assigned to one of the remaining three labels. Whilst labels positive and negative are self-explanatory, label complicated is applied when expressed sentiment is ambiguous or mixed. Custom-made distribution of data is shown in Table 2 and one can observe that, even though there are half as many tweets, the ratios are similar as in the Table 1.

For the purposes of this paper, labels are condensed into two labels: negative (NEG) and non-negative (NOT). Since this dataset originally explores the problem of complicated tweets, and our focus lies strictly on the application of our models to a different domain problem, we have taken into

account only those tweets which have over 80% consensus of the annotators. Also, to obtain a balanced dataset for testing, only a portion of the remaining dataset is chosen in order to have the same number of instances for each class. More specifically, the final test set consists of 600 tweets labeled as NEG and 600 labeled as NOT.

Table 2: Distribution of MTSA dataset

Label	Training	Test	Total
NEG	2,213	147	2,360
NOT	4,437	316	4,753
<b>Total</b>	6,650	463	7,113

#### 3.3. Pre-processing

For pre-processing the tweets several techniques are applied, such as normalizing the hashtags, mentions and elongated words, removing stop words and punctuation, converting words to lowercase etc. Described goal is achieved using standard tokenizers (Standard Core NLP), such as WordPunctTokenizer and RegexpTokenizer, as well as the NLTK TweetTokenizer<sup>1</sup> and TextBlob<sup>2</sup>.

### 4. Models

#### 4.1. Baseline Model

Simple baseline model goes through all of the words in a tweet and checks whether they appear in the dictionary of offensive words, which is constructed from three publicly available datasets<sup>3</sup>. Best results are attained whilst using the NLTK TweetTokenizer for pre-processing purposes, with accuracies being 0.705 and 0.701 for OLID and MTSA dataset, respectively. Interestingly enough, accuracy scores are similar for both offense and sentiment tasks, which indicates that tweets with negative sentiment often-times contain offensive words. This finding gave us the reassurance to pursue the testing on further, nontrivial models, and perform a comparison between the two tasks.

#### 4.2. Shallow Machine Learning Models

The shallow model used in this paper is logistic regression, with different features explored. For the first variant of logistic regression, features are extracted manually and are focused on negative connotation of words in a tweet or tweet itself. Ablation study was performed in order to decide which features should be used. Selected features are:

- Negative sentiment polarity - a single number extracted using VADER (Hutto and Gilbert, 2015) sentiment analysis system;
- Negative words - boolean feature indicating presence of words from the list of negative words (Hu and Liu, 2004; Liu et al., 2005);

<sup>1</sup><http://www.nltk.org/api/nltk.tokenize.html>

<sup>2</sup><https://textblob.readthedocs.io/en/dev/>

<sup>3</sup><https://www.kaggle.com/nicapotato/bad-bad-words>  
<https://gist.github.com/jamiew/1112488>

<https://github.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words/blob/master/en>

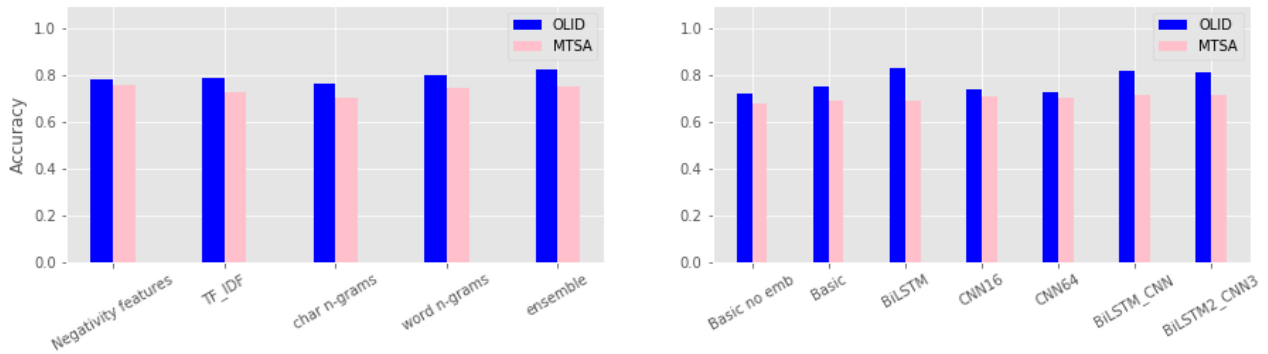


Figure 1: Comparison of different models for offensive language detection over 2 datasets: OLID and MTSA

Table 3: Performance of shallow machine learning models on OLID test set

Model	Accuracy	Precision	Recall	F1
LR - negativity features	0.779	0.711	0.358	0.476
LR - TF-IDF unigrams	0.791	<b>0.866</b>	0.296	0.441
LR - char n-grams	0.764	0.599	<b>0.491</b>	0.540
LR - word n-grams	0.798	0.770	0.404	0.530
Ensemble	<b>0.823</b>	0.848	0.467	<b>0.602</b>

- Bad/offensive words - boolean feature indicating presence of words from the list of bad words<sup>4</sup>.

Second variant contains unigram features weighted by the term frequency – inverse document frequency (TF-IDF). Last two variants use n-grams of different sizes weighted by the token counts as features. Second variant’s features are character n-grams of sizes from 3 to 5 and third’s are word n-grams of sizes 1, 2 and 3.

Finally, ensemble is built based on majority voting, whilst using only the hard predictions. Since there is an even number of models, both labels could get the same number of votes. To solve this problem, the predictions of the model which on average performed the best were taken into account twice. Only time when voting is not based on the majority is in the case when none of the words from a tweet have been recognized, which, most of the times, indicates that words have been misspelled (due to slang or human error). Therefore, in such occasions, the prediction of character-based variant is the only one used, which has been proven to perform well.

### 4.3. Deep Learning Models

For deep learning approach, we have explored various architectures. As the baseline, two simple dense-only models have been developed - BasicNoEmb, without pretrained embeddings, and Basic, with pretrained GloVe<sup>5</sup> 200d pre-trained embeddings.

All the other models have also been trained upon pretrained

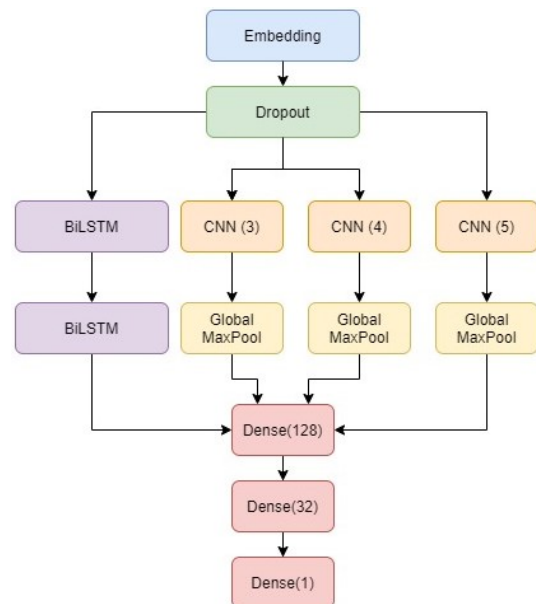


Figure 2: BiLSTM2\_CNN3 model architecture

GloVe 300d word embeddings. In Figure 1 (right), a few of those architectures are shown. Various combinations of Bidirectional LSTMs and CNNs are explored, in some cases used by themselves (but still combined with dense layers in the end), while in other cases LSTMs and CNNs are combined, with the intention of attaining better scores. The models shown in the Table 4 are BiLSTMs which consist of an embedding layer, BiLSTM layer and three dense layers. CNN models have a 1D convolutional layer and

<sup>4</sup><https://www.freewebheaders.com/bad-words-list-and-page-moderation-words-list-for-facebook/>

<sup>5</sup><https://nlp.stanford.edu/projects/glove/>

Table 4: Performance of deep learning models on OLID test set

Model	Accuracy	Precision	Recall	F1
BasicNoEmb	0.7223	0.529	0.512	0.502
Basic	0.752	0.564	0.571	0.550
BiLSTM	<b>0.829</b>	<b>0.785</b>	0.502	0.597
CNN16	0.741	0.545	<b>0.637</b>	0.569
CNN64	0.726	0.520	0.625	0.548
BiLSTM_CNN	0.821	0.694	0.612	0.633
BiLSTM2_CNN3	0.815	0.673	<b>0.637</b>	<b>0.644</b>

a global max pooling layer instead of BiLSTM. In Table 4, two versions can be seen - CNN16 with 16 filters and CNN64 with 64 filters as the output. Both CNN models have kernel size of 3 and a stride of 1. BiLSTM\_CNN model has a BiLSTM in parallel with a CNN of kernel 3 and stride 1, whose outputs concatenated together represent the input into 3 dense layers. Architecture of the most complicated model, BiLSTM2\_CNN3, is shown in Figure 2. This model consists of four branches whose concatenated outputs go into dense layers. In the first branch, we have two BiLSTM layers on top of each other, and in all the other branches are CNNs with kernel sizes of 3, 4 and 5 respectively.

## 5. Results

In search for a good model to represent an indicator of offensive language, a lot of models have been subjected to testing. Performance of which we can see in two tables: Table 3 provides results of shallow machine learning models and Table 4 of deep ones.

Using t-test it can be concluded, with great confidence ( $p < 0.001$ ), that all shallow models outperform the baseline model. The best shallow model is ensemble of models (t-test,  $p < 0.01$ ) which profits from all of the virtues of other variants. Because of LR char n-gram, ensemble can notice even misspelled swearword or some other word with negative connotation. It is also interesting to note that model with just three features (LR - negativity features) can achieve such results and become comparable with other, more complicated models.

Regarding deep models, it can be observed in Figure 1 that the models which include BiLSTMs tend to outperform slightly the ones that do not. It is noticeable that deep models perform differently based on their network structure. However, with 97,5% confidence we can say that all but two basic models outperform the baseline. We have found empirically, without an extensive parameter grid search, that pre-processing the tweets helps in the process of learning, that having less CNN filters often provides better or approximately the same score as having more CNN filters, and that usually our models struggle more with recall than with precision. In general, all of the models that include a BiLSTM perform very similarly.

One of our main questions is whether a model trained on the task of classifying offensive language can be used in a domain of classifying sentiment. As explained in Section

3.2., we have transformed the original MTSA dataset into a dataset which has two classes: "negative" and "non negative". Our assumption is that negative tweets will likely contain some of the words and constructions similar to ones contained in an offensive tweet, however sometimes tweets can be negative and non-offensive, thus models pretrained on OLID were expected to perform a bit worse on the MTSA dataset. As shown in Figure 1, the models indeed perform slightly worse on MTSA than on OLID dataset, but still they perform much better than a dummy classifier which would randomly assign labels to each input, in other words their performance is well over expected 50%.

We have explored some tweets for which the classifier is most certain (tweets whose model outputs are either lower than 0.1 which indicates that the model is quite sure in the label NOT, or higher than 0.9 implying the label NEG). One of the expected mistakes are tweets which are negative, but not offensive, such as "I had the meanest headache of life yesterday, fell asleep at 5 and just now waking up.. what is life". In contrast, in the example "My mom is going away tonight so imma have a sh\*t ton of coffee for dinner" is classified as negative, and its true label is non-negative. We assume that the model recognizes an offensive word ("sh\*t") inside the tweet and automatically labels it as offensive.

## 6. Conclusion

Nowadays, because of the extensive usage of social media, offensive language is becoming a great problem and detection of such behaviour is not a trivial task. In this paper, we propose a number of different models for solving the task of offensive language detection. On the one hand, from obtained results, it is obvious that some of the offensive language can be detected, either by extracting simple features such as presence of negatively connotated words, or by extracting more sophisticated features using pretrained word embeddings. On the other hand, recall values have shown that a portion of offensive tweets has still passed undetected.

The second goal of this paper is to examine whether models which classify offensive language are also able to detect negative sentiment. Described experiment is conducted on another dataset (MTSA) with models pretrained on OLID dataset. As expected, the results are slightly worse for sentiment analysis than for offensive language detection, but are considerably better than results obtained with a random classifier.

## References

- Thomas Davidson, Dana Warmusley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Eleventh international aaai conference on web and social media*.
- Cicero Nogueira dos Santos, Igor Melnyk, and Inkit Padhi. 2018. Fighting offensive language on social media with unsupervised text style transfer. In *56th Annual Meeting of the Association for Computational Linguistics*.
- Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. pages 168–177, 08.
- C.J. Hutto and Eric Gilbert. 2015. Vader: A parsimonious rule-based model for sentiment analysis of social media text. 01.
- Kian Kenyon-Dean, Eisha Ahmed, Scott Fujimoto, Jeremy Georges-Filteau, Christopher Glasz, Barleen Kaur, Auguste Lalande, Shruti Bhandari, Robert Belfer, Nirmal Kanagasabai, et al. 2018. Sentiment analysis: It’s complicated! In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1886–1895.
- Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: Analyzing and comparing opinions on the web. 05.
- Ping Liu, Wen Li, and Liang Zou. 2019. Nuli at semeval-2019 task 6: transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 87–91.
- Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153.
- Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.
- Gregor Wiedemann, Eugen Ruppert, Raghav Jindal, and Chris Biemann. 2018. Transfer learning from lida to bilstm-cnn for offensive language detection in twitter. *arXiv preprint arXiv:1811.02906*.
- Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval).

# How shallow are bots?

Gabriela Dorić, Ivica Duspara, Marko Kršić

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia  
{gabriela.doric, ivica.duspara, marko.krsic}@fer.hr

## Abstract

This paper presents a comparison between three models that are used to differentiate twitter bots from humans and female users from male users. The first (shallow) model tries to classify users based on the number of *hyperlinks*, *mentions*, *hashtags*, and *tweets* in the given dataset. The second (shallow) model is a simple NLP pipeline classifying the tweets via logistic regression, with the reasoning that women, men, and bots have a different vocabulary. Finally, the deep model is introduced - a simple CNN is trained in hope of improvement over the shallow models.

## 1. Introduction

According to cybersecurity company (Roberts, 2020) in 2019 around 37% of all traffic was bots. Bots which are harmless or perform helpful jobs (aggregator bots, backlink checker bots, etc.) are categorized as good bots. However, there is a rising trend of bad bots: in 2019 the percentage of bad bots has risen by 6% compared to 2018. Bad bots can perform various illegal activities ranging from choking bandwidth and scraping content and posting it elsewhere to orchestrating and executing coordinated attacks on services.

Social media is no stranger to bots. One such popular platform, Twitter, has an alarming problem with bots. In 2016 a study (Shu et al., 2016) has shown that 62% of adults in the US get news from social media. This can be troublesome because of misinformation and fabricated news (often called fake news) can be spread as truth. This coupled with a fact that a study from 2017 (Ferrara et al., 2017) shows that more than 48 million Twitter users are bots, means that differentiation between humans and bots on social media is a task worth pursuing.

In this work, we compare the performance of three models tasked with bot detection and gender profiling. The first model builds on the assumption that bots use more *hashtags*, *retweets*, and *mentions* than humans. The second model is a shallow one, represented by a simple NLP pipeline consisted of the TF-IDF vectorizer and some simple processing finished off by classification via logistic regression. The reasoning being that bots would use certain “buzzwords” frequently while humans wouldn’t and, similarly, female and male humans would express themselves differently. The third model is a deep learning model using simple CNN which uses tokenized blocks of text as input.

## 2. Dataset

Our dataset from PAN website<sup>1</sup>. The dataset is made up of an English and a Spanish sub-dataset. Each sub-dataset is divided into a training and a test set. Every data sample is represented by a pack of exactly one hundred tweets belonging to a certain Twitter user identified by the user ID.

<sup>1</sup><https://pan.webis.de/clef19/pan19-web/author-profiling.html?>

Table 1: Distribution of accounts in the dataset

	English		Spanish	
	Train	Test	Train	Test
Women	1030	660	750	450
Men	1030	660	750	450
Bots	2060	1320	1500	900

Every user is either bot or human, additionally, the human users are further divided by gender into male and female users. The distribution of data among classes is balanced for both tasks.

The training sets include 4120 users for the English sub-dataset and 3000 for the Spanish one while the sizes of test sets take up about 38% of according training sets. The dataset’s classes are perfectly balanced, and the dataset structure is described further in Table 1.

While there was a slight difference in mean tweet-length between a bot and human users in the English language – the same can’t be noticed in the Spanish dataset. However, there is a significant difference in the number of web-links and retweets averaged across users. Considering the English dataset, the bot users posted twice as many links on average and only a fifth of retweets relative to human users. In the Spanish dataset, the difference is a bit more subtle with bots posting about 60% more links and about a quarter of retweets compared to the Spanish human users. These stats are stable amongst training and test sets of the English version while the Spanish sets have some reassortments but having the general relations preserved. There are some differences between male and female users in average tweet length, but nothing worth noting, especially for the link postage and retweets.

### 2.1. Preprocessing

For the first model, raw data is preprocessed using a third-party library: *twitter-text-python*. For every user number of *hyperlinks*, *hashtags*, and *mentions* across all tweets are stored. These three numbers are features of the first model. The second and third model both share the first step in pre-processing. The stop-words are removed from the user’s tweet, all tweets of that user are concatenated into a string

Table 2: Average data in 100 tweets of the English training set

	Tweet Length	Urls	@	#
Bot	115.50	81.91	15.31	43.6
Male	93.82	38.23	101.54	24.48
Female	101.26	43.72	105.54	39.61
Human	97.54	40.98	103.54	32.05

Table 3: Average data in 100 tweets of the Spanish training set

	Tweet Length	Urls	@	#
Bot	92.15	59.65	35.46	51.06
Male	93.91	37.52	90.57	22.11
Female	86.69	31.23	90.67	25.92
Human	90.30	34.37	90.62	324.02

that represents user. The second model further vectorizes words using TF-IDF vectorizer and uses TF-IDF n-gram scores as weights. The third model applies tokenization to block of text content.

### 3. Design & evaluation

#### 3.1. First model

Social media bots in their interest have to spread and reach out to as many users as they can. With this in mind, when looking at Twitter this is achieved by users retweeting original tweets, using hashtags (which creates trends) and sharing links to attract users to specific websites. Furthermore bot accounts tend to tweet in one predictive pattern while human users are less predictive. The first model builds on these observations and makes them into features used for a model. These features are the *number* of the user’s tweets, *hashtags*, *retweets*, and *hyperlinks* across data of that user. Tables 2.-5. show average in 100 tweets.

As Table 6. shows, the first classifier achieves a decent score. It performs poorly on classification of genders.

#### 3.2. Second model

Another shallow model is applied to the problem. To enable the use of text (user’s tweet history) in a model, the tweets must be transformed into some form of suitable word representations e.g. vectors. These word representations are obtained via a TF-IDF vectorizer ignoring the letter case. The idea supporting the use of TF-IDF vectorizer is that bots exhibit a very spam-like behavior, recurrently using

Table 4: Average data in 100 tweets of the English test set

	Tweet Length	Urls	@	#
Bot	115.58	83.38	18.79	40.28
Male	96.44	44.23	100.93	32.05
Female	97.02	42.71	107.02	34.21
Human	96.73	43.47	103.98	33.13

Table 5: Average data in 100 tweets of the Spanish test set

	Tweet Length	Urls	@	#
Bot	96.45	62.27	22.64	60.71
Male	101.31	46.5	100.31	36.58
Female	87.91	41.81	96.1	27.39
Human	94.61	44.15	98.21	31.98

Table 6: F1 scores of the first model

	Bot detection	Gender detection
English	0.8078	0.4936
Spanish	0.7669	0.4779

the same words, usually “buzzwords”, that aren’t used by humans regularly. However, there are some bots that are outliers e.g. Bible reciter whose every tweet is generated by humans and differs from the rest of reciter’s other tweets, at least by its surface form (for this example, a classifier based on semantics might be more appropriate). Additionally, seen in Tables 2.-5., bots seem to post links a lot more frequently than human users. During the vectorization, to be examined are unigrams and bigrams of words (later referred to as “words”) – the addition of trigrams didn’t appear to affect the classification. The n-gram scores, i.e. weights, are computed as a combination of term frequency and the inverse document frequency, the latter used for re-weighting. The term frequency increases the word’s “importance” while the inverse-document- frequency decreases it, the idea being that if a word is frequent across many users – then it is not that classification meaningful. The IDF re-weighting appears to have similar effect to stop-words removal (including the dataset-specific stop-words, the words that have a low potential for differentiation). The TF-IDF scores inform on how “important” a certain word is to a user based on the frequency he used it. However, this kind of vectorization does not contain the user’s expressed semantics. The next pipeline step is the dimensionality reduction done by truncated singular value decomposition extracting only some components. Number of features (i.e. key n-grams) used is determined by a grid search, trying out combinations of number-of-features and the number of components after dimensionality reduction. About 50 features were optimal for every classification task with a reduction to about 30 components. Only a certain portion of best features is used, according to their TF-IDF scores. The post-dimensionality reduction remainder is normalized before the logistic regression is applied for binary classification (l2 norm is used for penalization).

The results of the classification can be seen in Table 7. This model performed the best on the task of bot detection among English users with F1 macro score of 0.89. Bot detection among the Spanish users had somewhat worse results with F1 score of 0.85 which could be influenced by the difference in the quality of English (higher quality, *Scikit-learn*’s built-in) and Spanish (lower quality, manually constructed out of unofficial lists) stop-words sets. Unfortunately, gender classification went as expected – an F1 score

Table 7: F1 scores of the second model

	Bot detection	Gender detection
English	0.8870	0.7206
Spanish	0.8527	0.6306

of 0.72 was acquired among the English users and, similarly, F1 score of 0.63 among the Spanish users. Removal of links from the tweets resulted in a decrease in F1 score of bot detection of about 2% in English and 1% in Spanish dataset while increasing F1 score by about 5% among English and decrease by 4% in Spanish. The link removal hasn't shown a stable effect across tasks.

It seems interesting to notice the features used for classification. In the English bot detection, the important features mostly regard things of a person's life-level significance such as *love*, *money*, and *job* while using very positive and promising words (e.g. *love*, *job*, *hiring*, *great*, *good*, *techjob*, *world*, *years*, *best*, *business*, *real*). Similarities can be seen in the Spanish dataset although in a bit softer fashion (e.g. *woman*, *love*, *life*, *years*, *house*, *need*, *millions*). The gender classification didn't go well for this model but some interesting gender differentiating features can be noted for both English (e.g. *love*, *hope*, *night*, *happy*, *life*, *look*, *think*, *time*) and Spanish set (e.g. *love*, *world*, *night*, *mercy*, *fault*, *thanks*, *god*, *photo*, *time*), again showing some similarity across languages. This kind of approach seems decent for the bot detection – the problem arises when there's a need for differentiating genders of human users. When classifying genders, the TF-IDF scheme does not produce satisfying results. Female and male users do appear to use different words and certain words in different frequency but not to the extreme of "bot vs. human" comparison. This can be seen by model's F1 score on gender training set of 0.72 (English) and 0.64 (Spanish), relatively low training score suggest that model just couldn't capture the problem (differences) properly (or hyperparameters weren't properly picked). Due to this, a deep model is introduced in the following section with the hope of providing better classification results.

### 3.3. Third model

To see if shallow models are enough for bot detection problem, we decided to compare it with a simple CNN model. The Model was built on top of Keras v2.3.1 and Tensorflow v1.15.0 and trained on GeForce RTX 2080 GPU.

#### 3.3.1. Preprocessing

To not lose data for training (Färber et al., 2019), considering we had merged tweets from one user into one unit, we set sequence length to  $L = 10000$  and applied padding to fixed-length  $L$ . That left us with 2880 examples for training and 1240 examples for validation.

#### 3.3.2. Model architecture

The Model architecture is presented in Figure 1. Input to our model is a matrix of dimensions  $d \times L$  where  $d$  is dimension of embedding vector for each word and  $L$  is the length of input sequence. To generate word embeddings we used pre-trained Google News corpus (3 billion running words)

Table 8: Hyperparameters of the third model

Conv. filter size	16,32,64
Conv. kernel size	2,3,4
MaxPooling1D pool size	2
Dropout rate	0.5
Dense layer units	256
Layers activation function	ReLu
Optimizer	Adam
Learning rate-adaptive	0.001 ->0.00001
Loss function	Binary cross entropy
Batch size	12

Table 9: F1 scores of the third model

	Bot detection	Gender detection
English	0.8999	0.4400
Spanish	0.6046	0.5201

<sup>2</sup> word vector model with dimension size  $d=300$  for each word. Model is compound of three one-dimensional CNN layers. To get features at different scales, for every convolutional layer we use a different number of features  $f$  and kernel size  $k$ . First, second, and third convolutional layer are defined with values  $(f=16, k=2)$ ,  $(f=32, k=3)$ ,  $(f=64, k=4)$  respectively. Every convolutional layer is followed by a dropout layer and Max Pooling layer. Dropout layers are used with a dropout rate of 0.5 in order to pick 50% random weights and set them to zero so that model would not overfit. Max Pooling layers are applied to extract only the most important features from CNN outputs. Max Pooling is done by applying kernel  $p$  of size  $p=2$  by which we reduce number of weights by two times. After 3rd convolutional layer, there is one fully connected layer with 256 hidden neurons. But to use the output from the last convolutional layer as input to fully connected layer, we applied a flatten method to reduce the dimensionality of convolutional output. The last layer of the model is another fully connected layer with just one neuron on which we applied sigmoid activation function to get final prediction (bot or human, male or female).

## 4. Statistical model comparison

As the dataset is split into training and test sets, the models are trained on the training sets and compared by their performance on the test set. The method of statistical comparison used is the permutation test. For each classification task, the models produce predictions of equal size. Models are compared in pairs. Firstly, the F1 scores are compared, followed by an iterative comparison of F1 scores achieved by shuffling the predictions randomly (10 000 iterations). The F1 score differences of shuffled outputs that exceed

<sup>2</sup><https://code.google.com/archive/p/word2vec/>

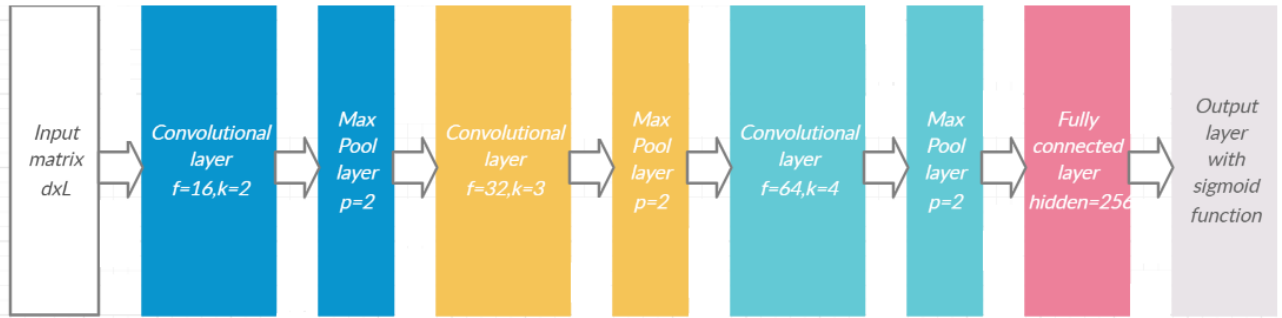


Figure 1: The architecture of CNN model

Table 10: Comparison of three models

Task	F1	Model 1	Model 2	Model 3
Bot - English set	Bots(1)	0.7743	0.8831	<b>0.8931</b>
	Humans(0)	0.8412	0.8908	<b>0.9067</b>
	Macro	0.8078	0.8869	<b>0.8999</b>
Bot - Spanish set	Bots(1)	0.7221	<b>0.8626</b>	0.5807
	Humans(0)	0.8117	<b>0.8426</b>	0.6286
	Macro	0.7669	<b>0.8526</b>	0.6046
Gender - English set	Female(1)	0.4436	<b>0.7337</b>	0.4099
	Male(0)	0.5434	<b>0.7074</b>	0.4701
	Macro	0.4935	<b>0.7205</b>	0.4400
Gender - Spanish set	Female(1)	0.3276	<b>0.5620</b>	0.4311
	Male(0)	0.6281	<b>0.6991</b>	0.6091
	Macro	0.4778	<b>0.6306</b>	0.5201

the original F1 score difference are counted and their ratio to the iteration number produces the p-val. The originally better scoring model was deemed significantly better if the p-val was valued under level of significance of 0.05. The first model was deemed the worst on all tasks and this test confirms it. The deep model was indeed the best one on the task of English bot detection but it was outperformed by the second (shallow) model on the rest of the tasks. Comparing the second and third models, it seems like the deep model had more focus on the single task while the shallow TF-IDF model, in its single form, could capture the user differences in a wider spectrum of tasks. Initially, we expected the deep model to perform far better on all tasks and possibly even outperform the shallow models. In further work, we'd inspect the deep model's low performance in the search of the cause.

## 5. Conclusion

We performed two classification tasks with three models. All three models have a decent performance on the bot detection task while they score lower on the task of gender classification. The first model could benefit from the inclu-

sion of information such as the number of followers user has and number of followed users. While doing error analysis on third model, we noticed an interesting phenomenon. When the deep model incorrectly classified humans as bots, 90% of these humans were males. This is a potential problem of fairness which is generally present in deep learning problems.

In future work, we plan to investigate this problem and apply a solution to our deep model. We concluded that bots, generally, aren't that deep. However, to detect bots to the maximum extent it does make sense to go deep. Unfortunately, the differences between genders remain foggy and a different approach is likely needed to yield better results.

## References

- Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. 2017. The rise of social bots.
- Michael Färber, Agon Qurdina, and Lule Ahmedi. 2019. Identifying twitter bots using a convolutional neural network.

Edward Roberts. 2020. Bad bot report 2020: Bad bots strike back. *Imperva blog*.

Kai Shu, Shuang Wang, and Huan Liu. 2016. Understanding user profiles on social media for fake news detection.

# SMS Spam Detection

Barthélémy MARSAULT, Florian GIGOT, Gaétan JAGOREL

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia  
{barthelemy.marsault,florian.gigot, gaetan.jagorel}@fer.hr

## Abstract

Nowadays, a phone number is almost part of the identity of his owner, a lot of legal and commercial forms ask for it as a mandatory information. So, the more a phone number becomes known, the more the owner becomes likely to receive spam. Furthermore, almost no phone provides a good and integrated spam filtering software. To fight this, some models already exist and in this paper, we present our specific implementations and concept that we tried to increase the precision of our filtering. It also goes through the way we tested and trained our work and some problems that we faced.

## 1. Introduction

SPAM or spamming are unwanted messages that you receive by email or SMS, having an advertising or a fraudulent goal and for which you didn't ask. Nowadays, communication via technologies as internet, email or mobile phone takes a large space in our daily life, so as this importance grows, the spamming grows too. Some say that the volume of spam is today up to between 60% and 90% of the daily traffic of email over the internet and spam has become a huge business for all kind of companies.

The problem with the spamming is that it's often annoying and aims to try to scam the recipient, it could be a simple advertising trying to force to buy something by an interesting ad, but it could also be phishing or trying to cause the download of a virus. So, now, a lot of personal information are managed on internet and accessible through a computer or a smartphone, we could consider the spamming detection as a cyber security threat. In fact, 64% of organizations have experienced a phishing attack in 2018 and the majority of this attacks were due to spams.

Regarding to the dangerousness that the spamming can involve whether for companies or private individuals, every respectful email boxes include a spam section and try to be as accurate as possible to differentiate spam from common messages. However, filtering options for mobile phone are not offering the same performances and are raising a lot of concern about the fact that an important message could be blocked. And even if the popularity of the SMS communication tends to decrease, the SMS spamming still appears to be quite popular in some region of the globe, like in China for example. Furthermore, now, a smartphone often stores bank, work or any other private data that can be stolen with simple hacking programs downloaded from a link in a spam SMS.

Knowing the possible issues link to the spamming, researchers are trying to solve the problem, but there are some difficulties in this field of research, for example, there is the lack of real public database of SMS. Besides, most of spam SMS come from unknown or changing phone number, which made them really hard to identify on this criteria. That only left the content of the message, which in most SMS, is short and fill with smileys and abbreviations.

To try to make a working spam detection system, we tried different existing models in order to compare possible results and get the best of them.

The rest of the article will include in second part the description of the dataset, then the process part talking about the semantic analysis, features and implementation. To end the paper, a brief part will explain the training of our system and the results that come out of it.

## 2. Dataset Description

The dataset we used is "The SMS Spam Collection v.1" created by Tiago A. Almeida and José María Gómez Hidalgo. This dataset is composed of 5574 real SMS messages in English, each tagged with either the Spam or Ham label. In this latter dataset, 86.6% of the messages are Ham and 13.4% are Spam. We used this dataset as a base to train and test our model. Moreover, it served as a reference for our statistical analysis of the differentiation features between spam and ham.

This dataset is the largest available, however it remains a small dataset for human language processing. Having a large dataset is very important to have enough data to train and thus increase the performance of the model. Moreover, a large dataset allows us to better test our model because we can have a larger test set (the data used for training and testing must be different).

## 3. Semantic Field Analysis

In this part, we will study the language used in the two types of messages: spam and ham. In order to know if a distinct semantic field differentiates the two classes.

First, a word frequency study is carried out for each type of message. If we look at the list of the twenty most frequent words, we find several similarities between the two lists. For example, the words "call" and "u" are found in both lists. However, the majority of the words are different, thus this path is promising. To explore this path, now let's look at the frequency of bigrams. A bigram is a sequence of two words like "I am". The list of the twenty most frequent bigrams of spam messages is very different from the list of ham messages. Moreover in these two lists, two bigrams stand out by taking the first place, "please call" for

Distribution between spam and ham

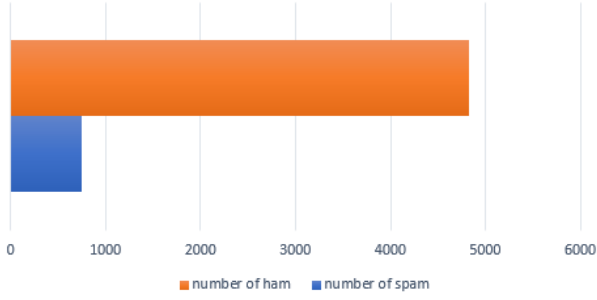


Figure 1: Shows the repartition between spam and ham messages in the dataset (747 spam vs 4827 ham).

Table 1: top 20 words in spam messages.

word	frequency
call	346
free	217
txt	156
u	144
ur	144
mobile	123
text	121
stop	114
claim	113
reply	104
prize	92
get	84
new	69
send	68
nokia	65
cash	62
urgent	62
win	60
contact	56
service	55

spam and "lt gt" for ham. Interesting fact, in both cases their frequency of appearance is much higher than the bigrams taking the second place, with a frequency respectively +184.5% and +475.9% higher.

To conclude, using the notion of bigram in our implementation could be interesting to increase the accuracy of our model.

#### 4. Feature Engineering

In this part, we look for features to differentiate between spam and ham. These features can be indicated to the model and thus support it in finding the correct solution. We studied the following features: the size of the message, the presence of a url link, the presence of a currency symbol and the presence of a phone number. In this list of features, only two are features that can assist the system to

Table 2: top 20 words in ham messages.

word	frequency
u	973
gt	318
lt	316
get	301
go	246
got	242
ur	237
ok	235
know	234
like	231
call	230
good	227
come	225
time	195
love	180
day	176
going	168
one	167
want	163
home	158

Table 3: top 20 bigrams in spam messages.

bigram	frequency
(please, call)	45
(po, box)	24
(guaranteed, call)	23
(prize, guaranteed)	22
(call, landline)	22
(selected, receive)	19
(contact, u)	19
(send, stop)	19
(every, week)	19
(await, collection)	19
(call, claim)	18
(urgent, mobile)	18
(call, land)	18
(land, line)	18
(customer, service)	17
(chance, win)	17
(free, entry)	16
(claim, call)	16
(private, account)	16
(account, statement)	16

differentiate the type of a message. These two features are the presence of a phone number and/or the presence of a currency symbol. Indeed, in our study only spam contains currency symbol and/or phone number. Thus, in our implementation we have applied pre-processing on the messages in order to clearly indicate to the model if these two features are present. To do this, we replace the number string

Table 4: top 20 bigrams in ham messages.

bigram	frequency
(lt, gt)	276
(gon, na)	58
(call, later)	50
(let, know)	39
(sorry, call)	38
(r, u)	37
(u, r)	35
(good, morning)	30
(take, care)	29
(u, wan)	29
(wan, na)	28
(lt, decimal)	23
(decimal, gt)	23
(new, year)	23
(u, get)	23
(pls, send)	22
(ok, lor)	22
(gt, lt)	21
(u, still)	19
(good, night)	19

representing the phone number with “phone-number” and the currency symbols with “money-symbol”.

### 5. Implementation

To implement the system we used python with the following libraries: pandas to get data set analysis management tools and sklearn to easily introduce machine learning models.

The structure of the system implementation is very simple (figure 2).

First we load a message, on this one we imply a first pre-processing to highlight the phone number and currency symbols if present. Then we imply a second pre-processing to enrich the message with the following information: are there any bigrams feature of spam or ham in the message ? Finally, the message is analyzed by a Multinomial Naive Bayes classifier (Shirani-Mehr, 2018) to determine whether the message is spam or ham.

### 6. Training And Evaluating The Model

To train our model we chose to use the first 4800 messages of our dataset, that is to say 86% of our dataset. The remaining 14% are used to test our model. Our objective being to have a maximum of data for training, while having enough for testing.

To evaluate our model, we used a total of 772 SMS which contains 671 ham messages (86.9%) and 101 spams (13.1%) . We based our evaluation on four criteria, the accuracy, the precision score, the recall score and the F1 score. We can see the results on Table 6.

Our system performed really well on the testing dataset with an accuracy of 99.09% which is quite impressive. In comparison to other models, our model outperform the

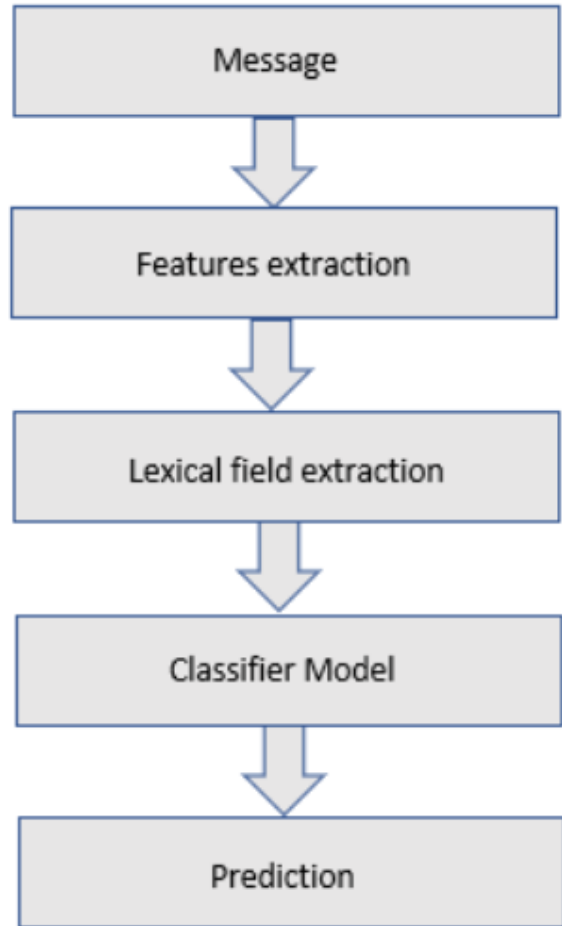


Figure 2: Procedure for implementation.

one created by Tiago A. Almeida, José María Gómez and Akebo Yamakami (Almeida et al., 2013) which was based on the same dataset and had an accuracy of 97.5%. However they used only the first 1674 messages for training their system while we used the first 4800 messages.

In our results, only 7 SMS were mispredicted as we can see in Table 5. We analyzed these messages in order to understand why the system failed. For example, the system predicted the message “Nokia phone is lovely..” as a spam message while it is a ham. This kind of messages is really hard to predict because it is really short and even for a human it is not easy to be sure if it is either a ham or spam. Another example of misprediction is “Hi this is Amy, we will be sending you a free phone number in a couple of days, which will give you an access to all the adult parties..”, our system detects this message as a ham but it is a spam. For this kind of messages, again it is hard to predict because there is no email address or phone number. Furthermore this SMS is written in a way that it looks like a normal conversation.

Finally, even if our results are very good, our system needs to be evaluated on a larger number of SMS to have better results.

Table 5: Detailed result of the prediction on the test set.

		predicted	
		ham	spam
actual	ham	688	3
	spam	4	97

Table 6: Predictions on the test set.

Accuracy of the model	0.9909326424870466
Precision score of the model	0.9820238095238095
Recall score of the model	0.977962550353396
F1 score of the model	0.9799809589431843

## 7. Conclusion

Automate the spam filtering is quite a hard task when it comes to SMS messages, during our work, we faced some important problems. The first one is the small number of representative SMS databases which made our system hard to test and train. The second one is the recurrent use of certain words on spam messages that often lead to a false positive detection on a ham. The third one is the fact that a lot of SMS use abbreviations and made them sometimes almost impossible to understand, which can lead to a misclassification.

As the result part shown, our results are correct, our work slightly improves the result of the original model we used and we can estimate that such an implementation could be helpful to avoid big stature scams. However, with the small number of spam examples that our system is trained on, an elaborate spamming technique could probably surpass our filtering.

Any future study on this domain (Hidalgo et al., 2012) should so considerate implementing bigram techniques to better work with the classifier.

## References

- Tiago A. Almeida, Tiago A. Almeida, and Akebo Yamakami. 2013. Contributions to the study of sms spam filtering: New collection and results.
- José María Gómez Hidalgo, Tiago A. Almeida, and Akebo Yamakami. 2012. On the validity of a new sms spam collection.
- Houshmand Shirani-Mehr. 2018. Sms spam detection using machine learning approach.

# Semantic Extraction from Cybersecurity Reports (SECR)

Markus Paulsen, Luis Fernandez, Bingji Wang

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia  
{markus.paulsen, luiz.fernandez, bingji.wang}@fer.hr

## Abstract

This paper describes a system called Semantic Extraction from Cybersecurity Reports (SECR), which utilises NLP pipelines, classifiers and lookup tables to automatically classify and annotate sentence relevancy and term labels as well as IT-security relevant effects and devices (Hard- and Software). This is done to simplify the task for IT-security researchers as well as software engineers to find information in existing APT reports, which might be important for their research field or developed software. The idea of the system is based on the SemEval-2018 Task 8 (Phandi et al., 2018) which only covers the automatic classification and annotation of sentence relevancy and term labels. The evaluation shows, that the developed system outperforms other baseline models, but it also uncovers some difficulties, which would be considered as part of a subsequent research.

## 1. Introduction

The young team around the German hobby hacker Karl Koch was in the late 80's probably not aware of the fact that their KGB-hack (Boulet, 2019) will later symbolise one of the schoolbook examples for a central topic of IT-Security: The Advanced persistent threat (APT). An APT can be seen from a space, time and computing perspective as a very complex developed attack, which targets a very specific and defined IT asset. Today, more than ever before, it is a main task of IT-Security researchers and software developers to identify and countermeasure APT's in order to avoid what Karl Koch has achieved: Massive damage. Another important field in today's IT area is the topic of Natural Language Processing (NLP), which can be found in search engines, advisor systems, automatic translation, document content analysis, ... So the research question for this paper is: How can the power of NLP be used to analyse APT reports for enabling IT-security researcher and software engineers to gain an insight of the current threats and the currently threaded hardware and software?

Exactly this question forms the foundation of this paper which is divided into 9 sections: The second (and next) section covers the work which was already done by other researchers in the fields of IT-Security and NLP. The third section explains the idea of a system, which we created to tackle the problem mentioned above, while the fourth section provides the reader of this paper with the necessary background knowledge. The fifth and sixth sections respectively describe the concept and the implementation of this system, while the seventh section provides an in depth evaluation of the systems performance. Finally in the last two chapters an outlook on future work and a conclusion is given.

## 2. Related Work

In the last decade, text mining approaches have started to be applied to IT-security research. The different proposals focus on detecting, extracting, and classifying vulnerabilities using NLP and machine learning models.

Various efforts have been made to apply NLP techniques to text-based malware reports. Author (Lim et al., 2017) proposed a dataset, labelling different malware capabilities, based on MAEC's (Kirillov et al., 2015) set of standard malware attributes. This was further developed during SemEval 2018 represented as a shared task on semantic extraction from cybersecurity reports (Phandi et al., 2018). A similar approach was taken by (Taneeya W. Satyapanich and Finin, 2020) creating a semantic model and including rich event annotations. Further information sources have been explored using unstructured data such as text from the web (Mulwad et al., 2011) or social media (Sabottke et al., 2015). This approaches try to construct models capable of predicting and extracting security threats.

Established techniques from information theory such as entropy and mutual information (Husari et al., 2018), information retrieval (e.g query expansion) (Khandpur et al., 2017) or topic modelling (Tsai and Chan, 2007) have been combined with NLP, trying to improve the results on topics from this field, requiring complex domain knowledge. Deep learning is the current state-of-the-art and architectures such as BiLSTM-CNN-CRF (Ma and Hovy, 2016) can be seen applied to solve tasks similar as the ones presented in this paper obtaining accuracy scores of 80.

## 3. Idea

In comparison to many other NLP based papers, our approach was not to just create another system for solving a well discussed problem, but trying to find a solution for a yet rather undiscussed problem by utilising the authors' findings while working on a more discussed problem. This more discussed problem is the SemEval-2018 Task 8 (Phandi et al., 2018), which asks contributors for a NLP based automatic relevance classification as well as for a NLP based automatic term, relation and attribute labelling of each sentence extracted from an APT Report (from a pool of APT Reports). Therefore we were provided by the authors of the task's paper with:

- Pre-relevance-classified APT Reports (.token files): Each .token file contains a list of tokenised words from the sentences of an APT report (e.g. 0 signalling an irrelevant word, all other signalling a relevant word)
- Pre-labelled APT Reports (.ann files): Each .ann file contains a list of annotated words from the sentences of an APT report. Among these annotations, there are term labels (indicating which role a word in a sentence has, e.g.subject, action, ...), relation labels (indicating relations between two term labelled words, e.g. subject for a action, object of a action, ...) and attribute labels (assigning certain actions to certain IT-security relevant effects, e.g. capability of an action).

We experimented with the provided data by loading it into a relational database and combining different parts of data via SQL. Hereby we found interesting relations between certain combinations of labelled subjects, actions and objects resulting in certain IT-security relevant effects and devices (hard- and software) (e.g. "attackers" (subject) + "access" (action) + "all the traffic to ad servers" (object) → "011:MalwareCapability-integrity\_violation" (effect)) or list of IT-security relevant devices (e.g. contains "Android services", "Java" or "multiple US Defense/Aerospace contractors").

Therefore our goal was then to create a system which:

1. automatically classifies and annotates sentence relevancy and term labels for an APT report (first two sub-tasks of the SemEval-2018 Task 8).
2. finds and annotates the IT-security relevant effects which the APT Report sentences trigger (by containing a certain combination of subject, action and object (or parts of it)).
3. finds and annotates IT-security relevant devices in each sentence of the APT Report.

For this goal it is necessary to train two classifiers (one for sentence relevancy and one for term labels) as well as to define two query statements. The first statement should link every action with its respective subject, object and effect to get a list of mappings [subject, action, object → effect]. The second one should gain every strategic object relation and link it with the respective object to get a list of IT-security relevant devices. Regarding the two classifiers, we tried to experiment with different state-of-the-art-classifiers in order to find the one with the best performance (especially a better performance compared to previous baselines).

## 4. Theory

The recent development of deep learning has facilitated the use of neural networks to solve downstream NLP tasks. Despite that the small size of the majority of datasets available for supervised NLP tasks resulted in an imperfect development of neural models for NLP compared to the results obtained in a parallel field such as Computer Vision. The emergence of the transformer architecture (Vaswani et al., 2017) and pre-trained language models has made it possible to easily approach NLP tasks from

a deep learning perspective. Self-Supervised learning allows the model to leverage huge unlabelled text corpus during the pre-training to first learn universal language representations and obtain contextual word embeddings. This is followed by supervised fine-tuning where the model is trained again on a smaller, labelled dataset relevant to the downstream task.

Different pre-training tasks and approaches result in different language models. Masked Language Modelling is used in BERT (Bidirectional Encoder Representation from Transformer) (Devlin et al., 2019) to obtain bidirectional language understanding. This is further improved with Enhanced Masked Language Modelling used in RoBERTa (Liu et al., 2019) by making the masking process dynamic. Relationships between sentences are learned using a Next Sentence Prediction task. The training of large-scale language models has been a recent concern (Sharir et al., 2020) and new approaches such as ELECTRA (Pre-training text encoders as discriminators rather than generators) (Clark et al., 2020) have been proposed, reducing the required computing payload, while maintaining its performance near the current state-of-the-art. Another problem has been the application of these huge models to low latency production services. A proposed solution is Distillation (Bucila et al., 2006) (Hinton et al., 2015) compressing the model while retaining its performance, exemplified in DistilBERT (Sanh et al., 2019).

## 5. Concept

The system, which we developed to tackle the problem mentioned in section 3., can be divided into two separate subsystems: The learning subsystem and the productive subsystem.

### 5.1. Learning subsystem

The task of the learning subsystem is to train the two classifiers mentioned at the end of section 3. Therefore the subsystem is divided into six components. The first component (called Document Service) stores all required files (.pdf, .token and .ann) so that other components can use these files. The next three components (Annotation Service, Token Service and APT Report Service) take the respective files (e.g. .pdf for APT Report Service) from the Document Service and transform them in respective data structures, which are easier to be stored in a database (e.g. transforms .pdf into the data structure APT Report).

The component Database Service then takes all the created data structures and stores their information into a database. Finally, the Classifier Learning Service component queries all required data and trains the classifier models with it. We evaluated different classifier models for the sentence relevance and the token label task and decided (based on performance data) to use the classifier RoBERTa (as described in Section 4.) for classifying sentence relevance as well as for token labels.

## 5.2. Productive subsystem

The task of the productive subsystem is to automatically classify and annotate sentence relevancy and term labels as well as to automatically find and annotate the IT-security relevant effects and devices as mentioned at the end of section 3. Therefore the subsystem is divided into seven components. The first component (called Database Service) stores APT report pages as well as token and term label data so that other components can query this data.

The Document Service component queries an APT report (disassembled into pages) from the Database Service and stores that data in an annotate-able data structure, called Document. The next 4 subsystems, called Relevance Service, Term Label Service, Effect Service and Device Service sequentially:

1. Classify the relevancy of every sentence in the Document.
2. Classify the term labels of every sentence’s word in the Document.
3. Query possible combinations of subjects, predicates and objects, which result in certain effects from the Database and search through the Document, to find whether such a combination appears in its text.
4. Query possible objects, which are related to a group of certain devices, using the same method as above.

All these results are respectively annotated on the Document data structure. Finally the User Interface Service component displays all the annotated data to the user.

## 6. Implementation

We implemented the system, described in section 5. in Python 3.7 with the support of the following external packages: PyPDF2 1.26 for extracting text from a .pdf file, NLTK 3.4 for separating that extracted text to sentences, simpletransformers 0.29 for creating and loading pre-trained models as well preprocessing the models input, wandb 0.8 for logging the training process and RxPY 3.1 for transforming the programming paradigm of Python from object oriented programming to stream oriented programming.

The full productive subsystem and most of the learning subsystem is realised in 17 Python classes and 1 SQLite database and is running on a local python instance, while the Classifier Learning Service is realised in 2 Jupyter notebooks running on Google Collab to utilise its GPU support, which significantly speeds up the learning process. The link between the two subsystems and especially between the Classifier Learning Service and the other components in the learning subsystem is the SQLite database, which represents, together with the stored classifier models, the persistent part of our system.

The classifier models are generated via the simple-transformer package in the Classifier Learning Service. The input of their learning process is generated via:

1. A single SQL query, which fetches the data from the database.
2. A preprocessing step, where the data format is adjusted.
3. A transformation into a DataFrame object via Pandas.

## 7. Evaluation

For the sentence classification task, three different models are introduced, namely ELECTRA, DistilBERT and RoBERTa, which all contain two classifiers (annotation classifier and sentence classifier). It’s worth mentioning that annotation should be done before executing sentence classification.

They are all tested by five standard APT report sets, which have been cleaned before testing and each contains dozens of APT reports. Furthermore, precision, F1 score and Matthews correlation coefficient (MCC) are chosen as system’s evaluation standards, calculated by their average performance in testing samples. Each testing sample is a complete APT report, thus method accuracy, recall and precision should be of the same value. Specifically, MCC is calculated by the formula below

$$MCC = \frac{k}{t} \quad (1)$$

where

$$k = TP \times TN - FP \times FN \quad (2)$$

and

$$\begin{aligned} s1 &= (TP + FP) \\ s2 &= (TP + FN) \\ s3 &= (TN + FP) \\ s4 &= (TN + FN) \\ t &= \sqrt{s1 + s2 + s3 + s4} \end{aligned}$$

TP is true positive, TN is true negative, FP is false positive and FN is false negative. The testing results of annotation classifier of the productive system are demonstrated in Table 1, where values are truncated to four decimal places. Both models perform well in evaluation, but RoBERTa works relatively better in both F1 score and MCC score and thus its annotation classifier is used as label classification in the system. As we use the whole APT reports as a testing sample, the precision and recall should be of the same value.

Methods	Precision	F1	MCC
ELECTRA	0.9129	0.9129	0.8800
DistilBERT	0.9194	0.9194	0.8879
RoBERTa	<b>0.9259</b>	<b>0.9259</b>	<b>0.8965</b>

Table 1: Performance results: Annotation results of three models of the productive system

Similarly, the sentence classifier is evaluated, but instead with only some parts of the annotated APT reports, the precision and recall are reintroduced. The evaluation

results are shown in Table 2, where RoBERTa is the best again among all standards. As a result, annotation classifier and sentence classifier are all selected from RoBERTa.

Although RoBERTa performs best in all the three models, it still shows low F1 as well as MCC scores in sentence classification, compared with the high scores in annotation results. The reason for this gap can be concluded as mistakes made in annotation have higher influence on sentence classification performance than annotation classification performance. For example, one mistake annotation made in an irrelevant sentence will push it to the relevant set, however, only lower annotation’s score is defined by a small value.

Methods	Precision	Recall	F1	MCC
ELECTRA	0.3621	0.7	0.4773	0.3841
DistilBERT	0.3693	0.7222	0.4887	0.4002
RoBERTa	<b>0.3825</b>	<b>0.7778</b>	<b>0.5128</b>	<b>0.4356</b>

Table 2: Performance results: Classification results of three models of the productive system

After the model has been trained and tested, newly produced unannotated APT reports are included to further test the model’s performance. If the sentences are annotated, the system will return as follows:

**Relevant sentence:** *Command and Control (C&C)*  
*The Dukes have employed several interesting tactics to hide the communications between the implants and their C&C servers, including the use of social media platforms and steganography.*

**Annotated sentence:** *Command(Subject), and(Modifier), Control(Action), (C&C)*  
*The(Subject), Dukes(Object), have(Action), employed(Action), several(Object), interesting(Object), tactics(Action), to(Modifier), hide(Action), the(Subject), communications(Object), between(Modifier), the(Subject), implants(Action), (Action), and(Modifier), their(Object), C&C(Object), servers.(Action), including(Action), the(Subject), use(Action), of(Modifier), social(Object), media(Object), platforms(Object), and(Modifier), steganography(Object).*

- 1) **First class effects:** None
- 2) **Second class effects:** None
- 3) **Third class effects:**  
 017: CommandandControl-recv\_data\_from\_c2\_server  
 018: CommandandControl-send\_data\_to\_c2\_server  
 093: Network-recv\_network\_packet  
 005: MalwareCapability-command\_and\_control  
 096: Network-send\_network\_packet
- 4) **Devices:** communications

Figure 1: Output sample

As shown in Fig.1, the system first outputs the original relevant sentence and then the annotated sentence, under which the IT-security effects and devices of this sentence are summarised. After all sentences are processed, the whole report’s IT-security effects are concluded as well. The example of IT-security summary is demonstrated in Fig.2, where the number of relevant sentences as well as all effects and devices in the report are included.

**Document summary:**

**Number of relevant sentences:** 27  
**First class effects:** None  
**Second class effects:**  
 1) 020: DataExfiltration-perform\_data\_exfiltration  
 2) 006: MalwareCapability-data\_exfiltration  
 3) 094: Network-send\_email\_message  
 4) 057: DataExfiltration-exfiltrate\_via\_network  
**Third class effects:**  
 1) 017: MalwareCapability-secondary\_operation  
 2) 013: MalwareCapability-persistence  
 3) 002: MalwareCapability-anti-detection  
 4) 183: System-add\_scheduled\_task  
 5) 092: InfectionPropagation-perform\_social-engineering\_based\_remote\_infection  
 6) .....

**Devices:** itself, targets, malware, shellcode, information, data, commands

Figure 2: Document summary

Usually, these new APT reports have not been cleaned and thus may contain unexpected or missing contents. The cleaning work needs to be done manually to remove the non-sentences (such as cover page, document header and links), and only keep complete sentences. The classifiers are still able to make the right decision on uncleaned sentences, as long as they contain the corresponding labels. However, in such a situation, errors usually occur when it comes to IT-security labels, because annotation classifier may assign labels to no-word contents.

## 8. Future Work

The used training and testing samples are manually cleaned APT reports from other researchers and including an automatic cleaning part will widen this system’s applications. Furthermore, it would be more accurate if the annotation is carried out by phrases instead by words. The gap between annotation classification and sentence classification is an important part which needs to be improved. In some cases, one sentence can produce dozens of IT-security effects, and finding the major effect can help people to understand the malware’s central danger. Therefore, four future work paths are proposed:

- Automatical cleaning model
- Annotate sentence with phrases
- Improving sentence classification performance and narrowing the gap to annotation classification.
- Evaluation and selection of IT-security effects

## 9. Conclusion

Compared with ELECTRA and DistilBERT, RoBERTa performs better in annotation classification and sentence classification, while the F1 and MCC scores of their sentence classifier are still unsatisfied, both being less than 0.6. Due to the high precision of the annotation classifier, the results of IT-security effects and devices are relatively reliable. The system is also able to work with uncleaned APT reports and can come up with right sentence classification if the sentence contains corresponding labels. In comparison, IT-security effect labels are not credible anymore as mistakes are more likely to be made in uncleaned sentences.

## References

Julie Boulet. 2019. The case of the cold war hacker found dead in a german forest.

- <https://worldcrunch.com/culture-society/the-case-of-the-cold-war-hacker-found-dead-in-a-german-forest>. Accessed: 31.05.2020.
- Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. volume 2006, pages 535–541, 08.
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *ArXiv*, abs/2003.10555.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.
- Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531.
- Ghaith Husari, Xi Niu, Bill Chu, and Ehab Al-Shaer. 2018. Using entropy and mutual information to extract threat actions from cyber threat intelligence. pages 1–6, 11.
- Rupinder Paul Khandpur, Taoran Ji, Steve Jan, Gang Wang, Chang-Tien Lu, and Naren Ramakrishnan. 2017. Crowdsourcing cybersecurity: Cyber attack detection using social media. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 1049–1057, New York, NY, USA. Association for Computing Machinery.
- Ivan Kirillov, Desiree Beck, Penny Chase, and Robert Martin. 2015. Malware attribute enumeration and characterization. 01.
- Swee Kiat Lim, Aldrian Obaja Muis, Wei Lu, and Chen Hui Ong. 2017. MalwareTextDB: A database for annotated malware articles. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1557–1567, Vancouver, Canada, July. Association for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.
- Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf.
- V. Mulwad, W. Li, A. Joshi, T. Finin, and K. Viswanathan. 2011. Extracting information about security vulnerabilities from web text. In *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 257–260.
- Peter Phandi, Amila Silva, and Wei Lu. 2018. SemEval-2018 task 8: Semantic extraction from Cybersecurity REports using natural language processing (SecureNLP). In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 697–706, New Orleans, Louisiana, June. Association for Computational Linguistics.
- Carl Sabottke, Octavian Suciu, and Tudor Dumitras. 2015. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, page 1041–1056, USA. USENIX Association.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.
- Or Sharir, Barak Peleg, and Yoav Shoham. 2020. The cost of training nlp models: A concise overview. *ArXiv*, abs/2004.08900.
- Francis Ferraro Taneeya W. Satyapanich and Tim Finin. 2020. CASIE: Extracting Cybersecurity Event Information from Text. In *Proceeding of the 34th AAAI Conference on Artificial Intelligence*. AAAI Press, February.
- Flora S. Tsai and Kap Luk Chan. 2007. Detecting cyber security threats in weblogs using probabilistic models. In Christopher C. Yang, Daniel Zeng, Michael Chau, Kuiyu Chang, Qing Yang, Xueqi Cheng, Jue Wang, Fei-Yue Wang, and Hsinchun Chen, editors, *Intelligence and Security Informatics*, pages 46–57, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

# Don't Worry, I Am Not Depressed... or Am I?

Luka Pavlović, Martin Sršen, Krunoslav Jurčić

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia  
{luka.pavlovic, martin.srsen, krunoslav.jurcic}@fer.hr

## Abstract

Language can provide powerful insight into personality, social or emotional status and also mental health. In this paper we researched which words contribute the most to classifying someone as depressive or someone as not depressive. On top of that, we tried adding several features that were extracted from the dataset to see if they might help make a better decision. Collection of data that we used is gathered from Reddit, a social media platform. It is comprised of posts and comments from control group and from group of people diagnosed with depression. For our model we chose logistic regression with term frequency-inverse document frequency (TF-IDF) because it showed to have the best performance. One of the features that provided a slight performance boost is average time between two sequential posts or comments. Our model showed better F1 performance than other models that used the same dataset.

## 1. Introduction

Depression is a common mental disorder whose importance is often being overlooked. If not treated properly, it can lead to more serious problems, disability, psychotic episodes and unfortunately in some cases even to suicide. According to World Health Organization, more than 264 million people worldwide suffer from depression. As it is the case with the majority of mental illnesses, early detection of the problem can prove to be very useful in its prevention. With the development of technology, people are spending more and more time on the Internet, and their activity on various websites (e.g. social networks, blogs...) can tell a lot about them. Language is a powerful indicator of personality, social or emotional status, but also mental health. It will surprise no one to learn that those with symptoms of depression use an excessive amount of words conveying negative emotions, specifically negative adjectives and adverbs such as “lonely” or “sad”. More interesting is the use of pronouns like “me”, “myself” and “I” which are used much more by those with symptoms of depression which means they are generally more focused on themselves. Someone’s social media account content can often provide us with valuable information about that person’s emotional status. However, the current technology used to deal with depression issues is only reactive. For instance, some specific types of risks can be detected by tracking Internet users, but alerts are triggered when the victim makes his disorders explicit, or when the criminal or offending activities are actually happening. We believe there is a better way in detecting early detection by using natural language processing tools which can be used on everyday social network posts. The main idea of this paper is to present one method of predicting whether a person is having depressive tendencies using natural language processing. We tried to achieve this by extracting words that are most commonly used by Reddit users who are suspected to have depressive tendencies alongside other features that proved to be group specific. The dataset used in this paper was first presented by Losada and Crestani in 2016. The dataset, which is described in more detail in Section 3, consists of numerous Reddit posts and comments made by various users. In order

to achieve that, we used a number of different approaches combined with different machine learning models, who are more accurately described in Section 4.

In this paper we make four main contributions:

1. we explore the influence of certain words or word phrases in detecting depression
2. we explore the influence of post/comment time frequency as a feature
3. we explore the influence of post/comment length as a feature
4. we explore the influence of sentiment in users posts as a feature

## 2. Related works

Even though depression is a well-known mental health issue, not a lot of datasets and papers issuing this distinctive natural language processing problem are present nowadays. This can be explained as a lack of general interest in the problem. Another reason is the fact that classification of such specific behavioural patterns can be quite challenging given the fact that the matter is rather subjective. Earlier mentioned paper by Losada and Crestani (2016) provided great insight regarding this topic. Their dataset is actually the first dataset for research on depression and language use. The details of this dataset, which was also used in this paper, are described in Section 3. For the evaluation of their model, a new evaluation metric was proposed by Losada and Crestani, called early risk detection error (ERDE) measure. This measure is based on the accuracy of the decisions and the delay in detecting positive cases.

Interesting paper which implemented sentiment analysis in order to assess whether the user has depression or not was done by Wang et al., where sentiment analysis method is proposed utilizing vocabulary and man-made rules to calculate the depression inclination of each micro-blog. Secondly, a depression detection model is constructed based on the proposed method and 10 features of depressed users derived from psychological research. We can see many researchers trying to develop models off of ever-growing social networks where a lot of text information is being placed. Using NLP with that kind of advantage to tackle

growing depression problem might be sufficient in years to come to explore depression further and provide psychiatrists with new and insightful data.

Another great approach was proposed by Benton et al. (2017) about estimating suicide risk and mental health in a deep learning framework additionally included the effect of modelling gender, which has been shown to improve accuracy in tasks using social media text. The authors of this paper developed neural multi-task learning (MTL) models for 10 prediction tasks (suicide, seven mental health conditions including depression, neurotypicality, and gender). The results of their model showed that choosing the right set of auxiliary tasks for a given mental condition can yield a better model, and that The MTL model dramatically improves for conditions with the smallest amount of data. Most importantly for our work, the results also showed that gender prediction does not follow the two previous points, but improves performance as an auxiliary task.

### 3. Dataset

The dataset used in this paper contains Reddit posts and comments from 892 different users. 137 users have been diagnosed with depression, and the rest are a control group. The maximum number of data for each user is limited to 1000 posts and 1000 comments which is Reddit API limit, and both the posts and the comments are sorted by chronological order, which is important given the task of early depression detection. The collection was created as a sequence of XML files, one file per user. Users are labeled as depressive only if they explicitly stated that they were diagnosed with depression and undepressed users are also from depression related subreddits but are not depressed, i.e. someone around them is suffering from depression so they want to explore it. Each submission is represented with:

- ID NUMBER
- TITLE
- DATE
- TEXT

The train-test split of the dataset consists of 486 train subjects, 83 of which are positive, and 406 test subjects, out of which 54 are positive subjects.

#### 3.1. Preprocessing

Cleaning data before using it to generate features for our model is very important, especially for Reddit data where we have several separated important groups of data. We first concatenated last N titles and texts for each user, where N determines how many posts our model requires to work correctly. Another type of data we extracted was time between each two of the sequential posts or comments for a certain user. After we grouped our data we then removed links from the text, lowercase everything and created “bag of words” representations of our sequences. Finally we applied WordNetLemmatizer on each word to create each word lemma.

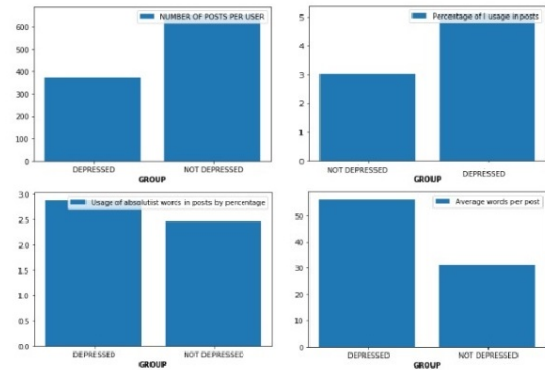


Figure 1: Data analysis on possibly interesting features mentioned in psychiatric literature

### 4. Problem

Studying depression related literature we found few claims that are supposed to characterise people suffering from depression and their interaction on social media. Claims we investigated are usage of 'I ..', absolutist words, negative sentiment in tweets, number of posts per user and time between posts. There is also one important observation considering depressed people on social media and that is time of posting, but those claims were already investigated in a paper by Banović, Fatorić and Rakovac from 2019, and proved not important enough, at least not on this dataset. Here you can see different characteristics and their behaviour in our data.

Analysis of our data showed that some assumptions really do show a solid difference between depressed and not depressed groups. Absolutist words (absolutely, totally, whole, ...) have no big difference between the groups, while average words per post and number of posts per user show significant difference.

#### 4.1. Model

We implemented several models that take into account the factors we mentioned earlier. Firstly we vectorize words using term frequency-inverse document frequency (TF-IDF), which is basically a statistic that determines how important a certain word is to a document in some collection or corpus. The TF-IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. For the task of classification we used a Logistic Regression classifier which proved to have excellent results on various text classification tasks. Those models were evaluated against baseline models. As baseline we used two different naive approaches:

- Random guesser : simple model which guesses whether a user has depression or not randomly. Each guess has the same probability of being chosen
- Stratified random guesser : another simple model with one small improvement; guess is not entirely random.

Table 1: F1 results from various methods

Model	F1 @ 5	F1 @ 10	F1 @ 20	F1 @ 50	F1 @ 100	F1 @ 200	F1 @ 500
RANDOM F1	0.1959	0.1959	0.1959	0.1959	0.1959	0.1959	0.1959
STRATIFIED F1	0.2060	0.2060	0.2060	0.2060	0.2060	0.2060	0.2060
LR TF-IDF	0.4375	0.5454	0.5903	0.64	0.6984	0.6871	0.6555
LR TF-IDF + SENT	0.3510	0.4984	0.5451	0.5981	0.6505	0.6395	0.6374
LR TF-IDF + POST LEN	0.3529	0.5012	0.5643	0.6334	0.6719	0.6570	0.6386
LR TF-IDF + SENT + POST LEN	0.3587	0.5043	0.5741	0.6195	0.6519	0.6471	0.6357
LR TF-IDF AVG DIFF BETWEEN POSTS	0.4375	0.5172	0.5669	0.6451	0.7086	0.6818	0.6722

Assumptions that proved insightful for our data and the features that we used to try improving model performance were: number of posts, sentiment of tweets, average time passed between posts and average words per post. Models were optimized following standard practices. We used grid search to fine tune models and TF-IDF vectorizer parameters. Parameters that showed to impact final results the most are the fact that we used both unigram and bigram words and that we ignored terms that have a document frequency strictly lower than the given threshold, the value which is in literature often called cut-off.

### 5. Results

Using this kind of model shows significant improvement over the F1 scores on the first 10, 100 and 500 posts over the model presented by Losada et al. This model also outperforms F1 scores across all identical data subsets by Banović, Fatorić and Rakovac. Although data analysis showed promising insights, the only feature that slightly improved base LR + TF-IDF model performance was average time passed between posts, while other additional features only created unnecessary noise that decreased models accuracy on the test set.

Important question is what can we learn from our models, and can we spot any pattern in detecting depression. Using the Logistic Regression model gives us the ability to extract words that have the most impact in determining which group a certain user belongs to. We've made word cloud out of those words to help visualize what model recognized as words that contribute classification the most.

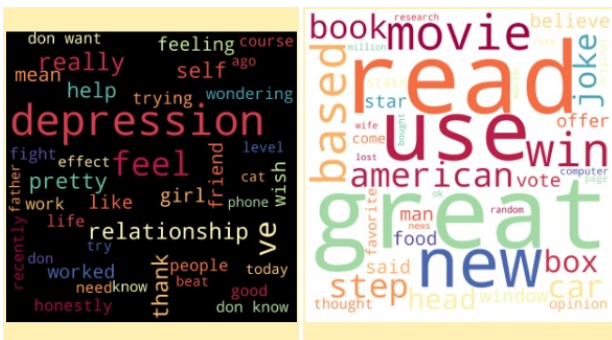


Figure 2: Unigrams that contribute most to depression classification (black background) and unigrams that contribute most to non depression classification (white background)

Solely on this unigram word clouds we can see that there are words that we would intuitively assign to depressed people so it's interesting to see how the model sorts words for both classes. We can also look up bigram words to see how well they reflect our intuition.

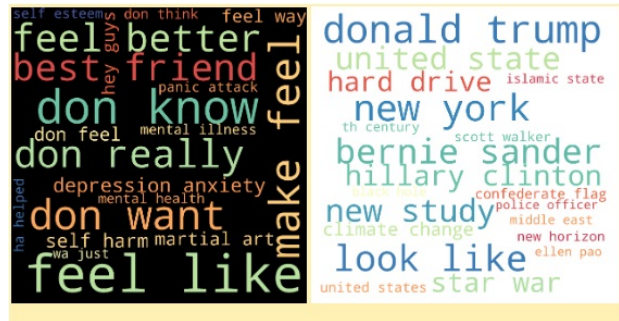


Figure 3: Bigrams that contribute most to depression classification (black background) and bigrams that contribute most to non depression classification (white background)

We can confirm from this data that those with some kind of depression are much more focused on themselves than on others.

### 6. Future Work

The biggest problem while creating this model was the size and content of the dataset. Increasing the dataset is a costly process but doing so would improve model performance. Time of posts, that was the only thing added alongside users' posts, helped us create a better model so it's safe to say that additional details on users could also benefit researchers. Work presented by Benton et al. in modelling gender as an additional feature could also prove useful. We might try using word embeddings averaged across all words as input to the logistic regression model or using it directly with Long Short Term Memory Networks which is the main ingredient of most state of art models since it can make use of long sequential input data and its ordering by avoiding gradient vanishing and making use of recurrent connections.

### 7. Conclusion

Depression is mental disorder that, if not treated properly and early enough, could lead to more serious problems or

even suicide. This kind of study could prove to be useful for psychiatrists, since a full history of posts can contain valuable insights to help better assess the patients. We experimented with the base model of the Logistic Regression model with TF-IDF features alongside other features that we thought could improve depression detection. Features we used were sentiment of posts, number of posts, length of posts, average time between posts where only average time between posts improved base model on certain data subsets. With that model we achieved a F1 score significantly higher than one presented in Losada et al. Our main goal was to find words or groups of words that contribute to detecting depression presented in section 5. Our work could be further improved by using deep learning models, including additional features that were not investigated in this paper such as gender of user or experimenting with different word embeddings.

## References

- Mohammed Al-Mosaiwi and Tom Johnstone. *In an absolute state: Elevated use of absolutist words is a marker specific to anxiety, depression, and suicidal ideation*. Clinical Psychological Science, 2018.
- W. Bucci and N. Freedman. *The language of depression*. Bulletin of the Menninger Clinic, 1981.
- David E. Losada and Fabio Crestani. *A Test Collection for Research on Depression and Language Use*.
- Adrian Benton, Margaret Mitchell and Dirk Hovy. *Multi-task Learning for Mental Health Conditions with Limited Social Media Data*.
- Luka Banović, Valentina Fatorić and Daniel Rakovac. *How Soon Can We Detect Depression?* Text Analysis and Retrieval 2019.
- Donik Vršnak, Mate Paulinović and Vjeko Kužina. *Early Depression Detection Using Temporal and Sentiment Features*. Text Analysis and Retrieval 2019.
- A. Genkin, D. Lewis, and D. Madigan. *Large-scale bayesian logistic regression for text categorization*. Technometrics, 49(3):291–304, 2007.

# Propaganda in Press: Challenges of Automatic Detection

Dora Pušelj, Tena Škalec

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia

dora.puselj@fer.hr, tena.skalec@fer.hr

## Abstract

Propaganda detection has been garnering more attention lately. In this paper, we deal with its two main challenges: identifying whether a fragment of text is propaganda or not and which type of propaganda technique is used. For identifying whether a fragment is propagandistic we experiment with the approach of disregarding context and achieve quite satisfying results. We compare several transformer models and find that BERT gives the best results. Furthermore, we tackle the issue of detecting different propaganda types by comparing the performance of the shallow models and BERT. Also, we deal with the problem of imbalanced dataset concerning different propaganda techniques. We experiment with several oversampling techniques used with BERT and achieve improvement in the detection of under-represented propaganda techniques, as well as the overall improvement of the performance.

## 1. Introduction

Artificial intelligence tasks are usually described as “easy for humans, difficult for computers”, but the point of propaganda is to go undetected as such when absorbed into our minds. How can a computer then be taught to spot the differences between honest text and one attempting to mimic it as best it can? We explore this in our paper, tackling a task similar to the one presented at the 2020 SemEval competition, titled “Detection of Propaganda Techniques in News Articles”.

We focus on two main subtasks: the first is to develop automatic tools for detecting whether given fragments of news articles are propagandistic (Span Classification), and the second is to determine the exact nature of detected propaganda, classifying it as one of fourteen established types (Technique Classification).

For the first task, we tested a number of models and settled on the implementation of BERT. We used the corpus of articles annotated by experts who marked fragments containing propaganda, extracting those fragments and separating them from snippets of honest text. We trained a classification model using roughly 10000 snippets and achieved good results.

As for the second task, we conducted two experiments. Firstly, we wanted to see whether there would be a significant performance difference between more traditional, shallow models and deep learning models (using BERT as an example). We concluded that BERT does indeed perform much better at the task of detecting different propaganda techniques. Secondly, we tackle the problem of the imbalanced data with several oversampling techniques used with BERT and find that they can manage to solve the problem quite successfully.

## 2. Related Work

Existing approaches to propaganda detection in news typically focus on evaluating an article as a whole and often take into account the publisher’s general character and credibility. The first automatic real-time propaganda detection system for online news was *proppy*, which estimated the to-

tal amount of propagandistic content in a given article and was developed in 2019. *Proppy* used four modules – for periodic article retrieval, event identification using DBSCAN clustering algorithm, deduplication and for propaganda index calculation (Barrón-Cedeño et al., 2019).

The research that inspired the 2020 SemEval’s propaganda task notes that discrepancies between the honesty of an article and its publisher are certainly possible, and therefore disregards the source when assessing articles. It also focuses on a more specific goal of detecting propagandistic fragments rather than grading articles as a whole, in order to identify propaganda explicitly (Martino et al., 2019).

The competition was just finishing up at the time of the writing of this paper; code submissions were closed and leaderboards published, but paper submission deadline was yet to come.

Consequently, few competing teams had published their work. The most notable amongst them was *ApplicaAI*, who achieved noteworthy results using systems based on the RoBERTa model and semi-supervised learning. In the first task, they utilized CRF layers, and for the second task, the team experimented with their own technique they called Span CLS which they describe as “roughly equivalent to adding consecutive layers and masking attention outside the span” of propaganda fragment (Jurkiewicz et al., 2020).

## 3. Dataset

The dataset created by the SemEval task organizers consists of about 550 news articles and the corresponding text files indicating article segments regarded as a type of propaganda. Annotated by a team of six experts, they differentiate 18 propaganda techniques (Martino et al., 2019). Even though there are overlapping spans in the articles, meaning that some fragments contain more than one propaganda technique, the task is treated as a multiclass classification problem. Since some of the techniques identified are quite rare they are merged into one superclass. For example, Bandwagon and Reductio ad Hitlerum are merged into “Bandwagon, Reductio ad Hitlerum”. Some are discarded completely, more specifically Obfuscation, Inten-

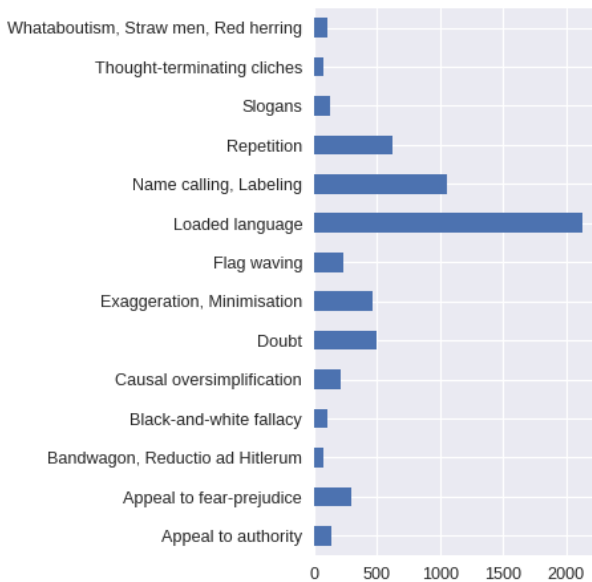


Figure 1: Class proportions in the training set

tional Vagueness and Confusion. Therefore, the task consists of 14 classes to be identified. Since we didn’t have access to gold labels for the test set, we used the development set for evaluation and will further refer to it as the test set in this paper. There are 6129 and 1063 propaganda fragments in the training and test set respectively, 7192 out of 11214 fragments in total.

Also, the dataset is quite imbalanced as shown in Figure 1, so it was important to tackle this problem in order to achieve satisfying results.

### 3.1. Text Preprocessing

Data was quite clean so preprocessing was minimal. For SC task, we simply split data into fragments labelled as propaganda or not and loaded them into dataframes. Cleaning them up further, splitting fragments that span multiple sentences, led to a decrease in accuracy, as did the removal of stop words, so we scrapped these changes. We suspect too much preprocessing ended up eliminating certain linguistic styles typical of propaganda which led to it being less distinguishable from honest text. Once propaganda was correctly identified, preprocessing data for the shallow models included automatic lowercasing and stop word removal. As for BERT, the pretrained model expects a certain input format, in which special tokens are used where [CLS] indicates the beginning of the sequence while [SEP] its end.

## 4. Our Approach

### 4.1. Task I: Span Classification

The first task was identifying whether a given fragment of text is propagandistic or not; we approached this as a binary classification problem.

Our corpus is composed of roughly 11000 text fragments from articles labelled as propaganda or honest using annotator-provided labels. We split those into two groups (propaganda, honest) to teach the classifier the distinction

between them. The main flaw of this approach is that the model is working with examples separated from context, consisting sometimes of a single polysemous word:

*Epidemic **appeared** to have been brought under control.*

The word *appeared* is propaganda here, meant to induce doubt in the reader, while the rest is factual information. In comparison, however:

*That lazy bum appeared **unbelievably late**.*

This example is an invert of the one above; it uses propaganda techniques before and after the predicate – name-calling and exaggeration respectively – while *appeared* is used as a neutral statement.

Such contradictions are possible and may be prevented by seeking and removing any duplicates with opposite labels – but this would have a negative impact on cases where a word appears once as propaganda and five times neutrally.

With this in mind, we ran tests on the corpus split into training and evaluation sets using four baseline models from SimpleTransformers library. The goal was to quickly estimate, using current state-of-the-art models, whether binary classification was a worthwhile approach taking the lack of context into account.

We also tested two possible methods of dealing with ambiguous data – if the model is uncertain how to classify a fragment, it defaults either to labelling it as honest (the “Benefit of the Doubt” approach, BD for short) or as propaganda (the “Better Safe than Sorry” approach, BSS for short).

The pretrained models from the Transformers library were chosen for a balance of speed and quality. The pretrained BERT is a 12-layer, 768-hidden-state, 12-head architecture with 110M parameters, trained on cased Wikipedia text. RoBERTa has 125M parameters with 12 layers, 768 hidden states, 3072 feed-forward hidden states and 8 heads and is trained on CommonCrawl data in 100 languages. The XLNet is a Large English model, a 24-layer architecture with 1024 hidden states, 16 heads and 340M parameters. The XLM English-German model is trained on Wikipedia and uses 6-layer architecture with 1024 hidden states and 8 heads.

We settled on BERT (Bidirectional Encoder Representations from Transformers), which has been very influential for the NLP community since it was presented by researchers at Google AI Language (Devlin et al., 2018). We experimented with fine-tuning its parameters and found the learning rate of  $7e^{-5}$  to show improvement over the default of  $4e^{-5}$ . Our experiments were quite restricted by hardware at our disposal, specifically the GPUs.

### 4.2. Task II: Technique Classification

The second task is a multiclass classification problem where a propagandistic fragment is given and needs to be identified as a certain propaganda technique. Several shallow models were trained for the task, including logistic regression, XGBoost and Linear SVM. They were chosen as they seem to work well regarding text classification. We also experimented with word representations and include the results achieved with tf-idf scheme and fastText embeddings.

Logistic regression, simple yet effective, was chosen as

one of the experimental models. The parameters were fine-tuned on the train data using grid search.

XGBoost is an ensemble model based on decision trees proven to be quite successful in numerous tasks (Chen and Guestrin, 2016). The parameters were fine-tuned using random search.

Linear SVM is widely regarded as one of the best text classification algorithms. Grid search was performed to fine-tune parameters.

Concerning deep learning models, we opted for a pre-trained BERT with a sequence classification head on top: essentially a linear layer on top of the pooled output.

### 4.3. Task II: Handling Imbalanced Data

We experimented with different approaches to oversampling, which is essentially generating new samples belonging to under-represented classes, to handle imbalanced data.

The simplest approach is random oversampling; randomly choosing samples from the minority classes and duplicating them in the new training set, with replacement. It can be effective but also cause overfitting, especially for higher over-sampling rates (Drummond and Holte, 2003).

Another method is ADASYN. It is based on the idea of generating synthetic samples from the minority classes based on how difficult they are to learn, using k-NN classifier. (Haibo He et al., 2008).

A similar approach is SMOTE, which doesn’t consider learning difficulty (Chawla et al., 2002). Its variant Borderline SMOTE (Han et al., 2005) generates new synthetic samples only based on the examples from the minority class near the borderline.

## 5. Results

Here we evaluate SC task in the first subsection, and the TC task in the second and third one. In the first subsection, we evaluate binary classification models using popular metrics. In the second subsection we compare shallow models and BERT, and in the third subsection, we compare different oversampling methods described in the Section 4.

### 5.1. Task I: Binary Classification

Initial tests were run three times each and the average scores are shown in Table 1. BERT achieved the best results overall, in terms of accuracy, F1-score and Matthews correlation coefficient (MCC). All measures represent confusion-matrix information, with MCC being regarded as particularly informative since it takes the balance ratios of all four categories into account (Boughorbel et al., 2017). Cross-entropy evaluation loss is displayed as well.

Of our tie-breaking approaches, “Benefit of the Doubt” (BD) proved slightly better for all models, though switching to the “Better Safe than Sorry” (BSS) with weighing biased towards honest text would also be a way to ensure the model doesn’t behave overly zealous in this regard.

We performed cross-validation to better evaluate our model on unseen data and achieved very satisfactory results shown in Table 2.

Table 1: Binary classification model scores.

Model	Approach	Loss	MCC	F1	Acc.
roBERTa	BSS	0.3180	0.8141	0.6405	0.9062
roBERTa	BD	0.3200	0.8185	0.6337	0.9093
BERT	BSS	<b>0.2430</b>	0.8467	<b>0.6416</b>	0.9231
BERT	BD	0.2450	<b>0.8613</b>	0.6361	<b>0.9306</b>
XLNet	BSS	0.4711	0.6808	0.6413	0.8314
XLNet	BD	0.3201	0.7964	0.6384	0.8974
XLM	BSS	0.4824	0.5550	0.5979	0.7775
XLM	BD	0.5164	0.6119	0.6070	0.8059

Table 2: 5-fold cross-validation accuracy, fine-tuned BERT.

	Acc.
Fold-1	0.9741
Fold-2	0.9732
Fold-3	0.9674
Fold-4	0.9710
Fold-5	0.9759
Mean:	0.9724
Standard deviation:	0.0029

### 5.2. Task II: Shallow Models and BERT

Here, we compare the results achieved with various shallow models and BERT. The achieved micro F1-scores for each model evaluated on the test set are presented in Table 4. For the sake of fairness, we include two versions of BERT: one trained with no oversampling method applied, and one which utilizes Borderline SMOTE oversampling. We chose logistic regression trained with tf-idf scheme as the baseline and we use permutation testing at the 95% significance level to compare the different models’ performances.

Logistic regression trained with fastText embeddings and Linear SVM trained with both tf-idf scheme and fastText embeddings shows no significant performance gain over the baseline. Interestingly, XGBoost model trained with tf-idf scheme is outperformed by the baseline significantly ( $p = 0.0001$ ), while the one trained with fastText embeddings outperforms the baseline ( $p = 0.0447$ ). Moreover, both BERT models show significant performance gain compared to both the baseline and the XGBoost trained with fastText embeddings ( $p < 0.0001$ ). Thus, we conclude that the BERT’s superior performance is not just due to chance. All in all, BERT seems to be more well-suited for this task than the presented shallow models.

In the next subsection, we analyze different oversampling methods applied to BERT. It should be mentioned that for this purpose we only consider BERT because applying these methods to the shallow models gave us rather poor results, and thus, we decided not to focus on them.

### 5.3. Analysis of Different Oversampling Techniques

The results of the experimenting with different oversampling techniques can be found in the Table 5; they include micro-F1 and macro-F1 scores for BERT in respect to the oversampling technique applied, alongside the BERT

Table 3: F1-scores for several propaganda techniques in respect to different oversampling techniques and the number of samples from each class in the train and the test set (the columns named test and train) – with BERT as a model.

Oversampling technique	None	Random	SMOTE	Borderline	ADASYN	Test	Train
Loaded language	<b>0.7768</b>	0.7084	0.7663	0.7688	0.7270	325	2123
Black and White Fallacy	0.0000	0.1379	0.1935	<b>0.2143</b>	0.1875	22	107
Bandwagon, Reductio ad Hitlerum	0.0000	<b>0.4000</b>	0.0000	0.1667	0.0000	5	72
Name Calling, Labeling	0.7059	0.6461	<b>0.7328</b>	0.6933	0.7024	183	1058
Exaggeration, Minimisation	0.4605	<b>0.5200</b>	0.4324	0.4636	0.3837	68	466
Doubt	0.9185	0.8794	<b>0.9538</b>	0.9313	0.9394	66	492
Appeal to Authority	0.0000	0.0667	0.2500	<b>0.3158</b>	0.2222	14	142
Repetition	0.3465	0.2110	0.4143	<b>0.4631</b>	0.3568	145	620

Table 4: Micro-F1 scores on the test set.

Model	micro-F1
Logistic regression + tf-idf	0.4675
Logistic regression + fasttext	0.4854
Linear SVM + tf-idf	0.4817
Linear SVM + fasttext	0.4797
XGBoost + tf-idf	0.4243
XGBoost + fasttext	0.4958
BERT (no oversampling)	0.6237
BERT (Borderline SMOTE)	<b>0.6500</b>

Table 5: Comparing oversampling techniques on BERT.

Oversampling technique	micro-F1	macro-F1
None	0.6237	0.3540
Random	0.5974	0.4332
SMOTE	0.6444	0.4501
SMOTE Borderline	<b>0.6500</b>	<b>0.4808</b>
ADASYN	0.6209	0.4432

model with no technique applied – which we will further refer to as the “plain” BERT. We include both micro-F1 and macro-F1 because we think it’s important to see the differences in both scores since micro-F1 is more forgiving towards the classes that are in minority, while macro-F1 treats every class the same. Usually, which score is preferred to evaluate the model depends on the task that needs to be addressed.

By looking at the results, we can observe that while there are obvious differences in both scores, there is a more dramatic increase in the macro-F1 score when oversampling techniques are applied. Interestingly, while Random oversampling achieves lower micro-F1 score, it achieves higher macro-F1 score. This is likely due to the model predicting the under-represented classes better at the expense of the more represented ones.

To see whether the differences achieved are significant, we perform permutation testing at the 95% significance

level (like in the experiment with the shallow models) with the plain BERT as the baseline. Considering micro-F1 scores, the plain BERT model outperforms the Random oversampling one. All other models achieve better micro-F1 scores than the plain BERT, but only the Borderline SMOTE version outperforms it significantly ( $p = 0.0467$ ).

Furthermore, when considering macro-F1 scores, all the models that use oversampling techniques outperform the plain BERT by a significant amount. This experiment leads us to believe that in this case there is an obvious advantage to using oversampling methods in contrast to using none at all, especially in the case of the Borderline SMOTE oversampling technique.

For the sake of a more detailed analysis, we compare a number of propaganda techniques according to the oversampling method (and the plain BERT) – considering only the specific propaganda technique. This is shown in Table 3, where F1-scores for some propaganda techniques and oversampling method is shown. Some of the techniques that have been predicted correctly zero times with plain BERT achieve better scores with prior oversampling.

## 6. Conclusion

In this paper, we discussed the challenges of propaganda detection in news articles and experimented with several ways to approach the issue.

We analyzed the performance of several transformer models we put to the task of identifying propaganda snippets in articles and found that BERT achieved the best results overall. The quality of results was especially interesting considering we worked with snippets of text taken out of context which their meaning sometimes depends on, of varying length and sometimes spanning across sentences.

In addition, we have tackled the task of detecting different propaganda techniques. We compared several shallow models with BERT and found that BERT significantly outperformed other models. Also, we experimented with different oversampling techniques and concluded that they result in performance gain when used with BERT.

## Acknowledgements

We would like to extend sincere thanks to Professor Jan Šnajder and Assistant Professor Mladen Karan for all the guidance and knowledge we acquired.

## References

- Alberto Barrón-Cedeño, Israa Jaradat, Giovanni Martino, and Preslav Nakov. 2019. Propy: Organizing the news based on their propagandistic content. *Information Processing Management*, 56, 05.
- S. Boughorbel, Fethi Jarray, and Mohammed El-Anbari. 2017. Optimal classifier for imbalanced data using matthews correlation coefficient metric. *PLOS ONE*, 12:e0177678, 06.
- N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, Jun.
- Tianqi Chen and Carlos Guestrin. 2016. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.
- Chris Drummond and Robert Holte. 2003. C4.5, class imbalance, and cost sensitivity: Why under-sampling beats oversampling. *Proceedings of the ICML'03 Workshop on Learning from Imbalanced Datasets*, 01.
- Haibo He, Yang Bai, E. A. Garcia, and Shutao Li. 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328.
- Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In De-Shuang Huang, Xiao-Ping Zhang, and Guang-Bin Huang, editors, *Advances in Intelligent Computing*, pages 878–887, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Graliński. 2020. Applicaai at semeval-2020 task 11: On roberta-crf, span cls and whether self-training helps them. 05.
- Giovanni Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news articles. 10.

# Search Less, Research More: Improving Information Retrieval in the Face of COVID-19

Mario Šaško, Ivan Lovrenčić, Nikola Buhiniček

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia  
{mario.sasko, ivan.lovrencic, nikola.buhinicek}@fer.hr

## Abstract

As the persistent COVID-19 outbreak transforms the world, scientists are vigorously researching the virus to find a cure. For researchers to stay updated with the latest developments, they have to find the time to evaluate current advancements and potentially apply them in their research. However, because of the global initiative to solve this crisis, new research is getting published continually. That causes an abundance of research papers, where most of them do not deliver any significance to the on-going cure development. To help researchers find the most prominent improvements, we propose the solution in the form of an information retrieval model, that we adapted to the scientific domain of this problem. Our model will allow researchers to find the most relevant papers for each of their COVID-19 related queries, and in the process, save some valuable time.

## 1. Introduction

In the last days of 2019, numerous reports about the previously-unknown virus have appeared in Wuhan, China. Not even a few weeks after that, the World Health Organization (WHO) started alerting about a seeming global health crisis. Six months later, the highly-infectious novel Sars-Cov-2 virus has infected more than five million people and more than 340,000 people have died. Furthermore, researchers believe that this emergency will not cease until there is a viable cure, and, naturally, as a part of the global response to find it, researchers are publishing more papers than ever. However, from that arose another challenge. There is now an abundance of research papers being published, and researchers are keen to find a way to retrieve only the most prominent ones to save some precious time.

To help researchers, the White House and the coalition of leading research groups came up with an initiative to prepare the COVID-19 Open Research Dataset (*CORD-19*).<sup>1</sup> This accessible dataset was given to the global research community to apply various natural language processing (NLP) and machine learning (ML) techniques. These methods could help scientists find the most relevant research about the on-going pandemic and, consequently, give us a better understanding of the Sars-CoV-2 virus.

To contribute, we have decided to develop our IR model. The main intention of our approach was to allow researchers to obtain the most relevant papers based on their query input. To achieve that, we have used a stratified model, that through each level, filters and ranks papers based on their query relevance. The model starts with a biomedical named entity recognition (BioNER) to filter out papers that are not related to the current query. Furthermore, the model proceeds by utilizing the word (word2vec) and document (doc2vec) level embeddings to determine the remaining papers significance rank. In the end, we are left with a list of query-related papers, which are sorted based on their relevance score.

<sup>1</sup><https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge>

In the following section, we will discuss the approaches of other highly-ranked proposed solutions and compare them to our own. Furthermore, in Section 3., we will explain how we used the given CORD-19 dataset for evaluation of our model, which will be explained in detail in Section 4. Moreover, in Section 5., we will talk more about evaluation and results. We had challenges in the evaluation part, as it was hard to evaluate the significance of the research papers from the domain we are not experts. Lastly, we finish with a conclusion and ideas for future work.

## 2. Related Work

The mentioned CORD-19 dataset came along with a series of coronavirus related tasks and so far there has been over 1,400 submissions. The task we have chosen is related to the queries about the transmission and incubation of the novel Sars-Cov-2 virus.

In that task, the officially accepted *submission*<sup>2</sup> is using only paper abstracts for relevance determination. Firstly, the query is being preprocessed into keywords, which are then stemmed and concatenated with COVID-19 related terms (covid, cov, -cov, hcov). Furthermore, they filter the papers by checking whether the abstract contains all of the queried keywords and at least one COVID-19 related term.

In most of the remaining submissions, researchers calculated the relevance scores based on the keyword counts and the length of only the abstract part. However, based on our research and the impressions we encountered, the decision to use only abstracts is not justifiable and is rather wasteful. Our model utilizes more of the available data by weighting parts of papers differently, and not relying just on the short summaries.

On the other hand, we have a highly rated probabilistic *approach*<sup>3</sup>, where researchers often used a BM25 model, along with document embeddings. In those kinds of sub-

<sup>2</sup><https://www.kaggle.com/mlconsult/transmission-incubation-and-environment-2-0>

<sup>3</sup><https://www.kaggle.com/dgunning/cord-research-engine-search-and-similarity>

mission, they use the BM25 search index, created only from paper abstracts, as their base and boosting the performance with an Annoy index made out of 786 dimensional Specter Vectors (Cohan et al., 2020), which represents document embeddings. In contrast to those submissions, we have decided to utilize both word and document embeddings to get an even better insight.

Lastly, although our model does not explicitly utilize any of the ML tools, we thought to mention one of the approaches that accomplished notable results. In the stated paper, the researchers relied more on the article body, which they parse using various NLP tools. Furthermore, they turned document into a TF-IDF based vector, on which they apply the k-means clustering. In the end, they managed to cluster papers based on some surprising factors that could potentially shed more light on this mysterious virus and the data that we have about it. However, the approach itself did not produce a state-of-the-art result.

### 3. Dataset

As mentioned in Section 1., we use the CORD-19 dataset to evaluate the model. The dataset contains over 134,000 scholarly articles with approximately half of them being related to COVID-19. Out of the total number, 29315 articles are presented as a full-text JSON. In the task, we focus on fields title, abstract and body and discard the rest, which aligns with our assumption that these fields are the most important ones.

In order to measure the model performance, we use a simple random sampling without replacement to sample the dataset and obtain 95 articles. Next, since we use 2 queries to evaluate the model, for each query separately, each article is annotated as relevant or irrelevant by 3 different annotators. Finally, the majority agreement is used to define gold-standard labels.

During preprocessing, the aforementioned JSON fields are concatenated and tokenized using *spaCy*<sup>4</sup>, an advanced natural language processing library. Then, the input is lowercased and passed to the model. The same steps, with additional removal of stop words, are applied when working with the baselines.



Figure 1: System architecture.

## 4. Our Approach

For our final model, we decided on a stratified structure model. We aimed to make a system where we would have the ability to add and remove layers without a lot of effort. Moreover, we wanted to have a continuous flow of data through the model, allowing us to evaluate the model at each layer. Figure 1 nicely illustrates the layered structure we have applied.

<sup>4</sup><https://spacy.io/>

Beside distinctive system structure, our approach differs from other submissions in another regard. Specifically, we utilized every part of the paper, as opposed to most, where they used only the abstract. In our experiments, we have noticed that having additional data, like title or body, can make a significant difference. Consequently, we have implemented a system where each part of the paper will be treated individually.

As we can notice in Figure 2, we split our input data into three parts. Then, we pass those parts through each layer of our model. Furthermore, we sum the individual scores and calculate the average. Now, it should be noted that we repeat this process at each layer and not only at the end of the model. We use this output as the customized relevance score, which we are updating with every layer. Lastly, the model completes with a sorted list of papers, based on the mentioned customized relevance score.

In the following sections, we will explain in detail our baseline models and every individual layer in our final model. Moreover, we will describe which internal parameters we used and how we optimized them for this specific task.

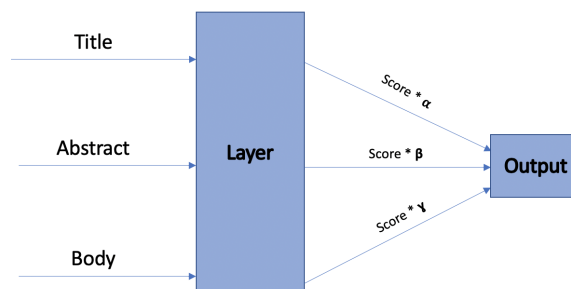


Figure 2: System architecture where we score each paper based on its components.

### 4.1. Baseline

We use two baselines to validate the performance of our model. Both models are part of *Apache Lucene*<sup>5</sup>, a high-performance search engine library.

The first one of them, an unigram Language Model (LM), builds one probabilistic model per article in collection. Then for each article, the model outputs the probability of a query belonging to it. We sort these probabilities to obtain the final relevance ranking. To make predictions more robust, the model relies on Bayesian smoothing with Dirichlet priors (Zhai and Lafferty, 2004).

As our second baseline model, we use a BM25 Okapi ranking function (Robertson et al., 1995). The ranking function weighs query terms according to an advanced TF-IDF weighting scheme. Since this is the only function from BM family that properly scores longer articles, we don't consider other variants. Additionally, the same sorting procedure that is used with the LM model is applied to the function output.

<sup>5</sup><https://lucene.apache.org/>

## 4.2. BioNER

BioNER is a robust NLP tool that acquires solely biomedical phrases from the given document (Wang et al., 2019). In our case, we applied it to extract the relevant biomedical keywords from queries and papers. Once we obtained the keywords, we matched query keywords to the corresponding keywords from title, abstract, and body. However, we would not treat every match identically. Specifically, as we can notice from Figure 2, the final scores are multiplied with a specific hyperparameter. Those hyperparameters were optimized specifically for this particular task, as they were used to give advantage to more significant parts of the paper.

Moreover, we also applied BioNER to obtain precise keywords that should not be in the relevant papers. For example, some papers from the dataset are referring to the Sars-CoV-1, the past virus that caused an epidemic in 2003. In these situations, we want to find those phrases and appropriately correct the paper’s relevance score. Without this, it was effectively impossible to differentiate between relevant and previously relevant papers, as those past papers usually include a great deal of currently relevant keywords.

## 4.3. Word2vec

As a complement to the previous layer, in this layer, we use word embeddings to find a semantic relation between the query and the document (Mikolov et al., 2013). Consequently, if the paper in the previous layer got a high score, but it is not semantically similar to the query, it will receive a low score in this layer, which will decrease its relevance score. This reiterative score recalculation allows us to find the most suitable papers, based on more than one approach.

To determine the semantic similarity between them, we add each word’s vector representation and then compute the cosine similarity between the query and the paper. For this part, we have applied a pre-trained Google’s word embeddings. As stated, in each layer of our system, we calculate scores independently for each section of the paper - title, abstract, and body. In this case, we add up all the calculated cosine similarities, and we take an average value.

## 4.4. Doc2vec

In the context of the on-going pandemic, word2vec has one prominent impediment. Namely, the issue stems from the fact that most of the terminology from the current pandemic is new. Consequently, most of those new terms, like COVID-19, are not in the word2vec vocabulary, hence the overall accuracy of word2vec representations will be worse. To solve that, we added a layer of document embeddings (doc2vec).

To build vector representations of individual papers, we have to train the doc2vec model on our dataset (Le and Mikolov, 2014). However, in our case, the model is exclusively trained on a subset that the prior layers listed relevant. In other words, vector representations are produced only for documents that have been named relevant by BioNER and word2vec layers. Therefore, this layer is a reinforcement layer, that will adjust the ranks, and potentially correct any irregularities in the relevance scores.

## 4.5. Hyperparameters

As previously stated, our model performance massively relies on three hyperparameters. We added those parameters because we noticed that specific sections of the paper bring much more weight than the others.

First, we discerned that matching keywords in a title considerably increased our chances of guessing the relevant paper. However, papers that contain query words in the title are rare. Therefore, we added two more hyperparameters that would reward matched keywords in the abstract and the body.

After numerous optimization attempts, the hyperparameters converged into the following values:  $\alpha = 2.5$ ,  $\beta = 1.5$  and  $\gamma = 1$ . As we can see, the title’s hyperparameter has the biggest value, which implies that the title carries the most significant amount of information about the relevance of the paper. Sequentially, we see that the abstract is second and that words from the body carry the least amount of weight. We optimized hyperparameters by relying on our judgment of top  $K$  papers that were retrieved. We settled with this empirical evaluation because we did not have time and resources to perform cross-validation on the annotated dataset.

Lastly, we are going to discuss how to calculate relevance scores. Namely, there is a discrepancy in how we estimate relevance scores between layers and inside the individual layer. Specifically, as we can see in Equation 1., the relevance score in each layer is calculated by dividing the sum of individual parts by the factor of 3. Meanwhile, to calculate the new relevance score between layers, we average the current score and the score from the previous layer.

$$Relevance = \frac{title * \alpha + abstract * \beta + body * \gamma}{3} \quad (1)$$

## 5. Results

As mentioned in Section 3., two COVID-19-related queries are defined to evaluate the model. For each query, the model assigns relevance scores according to which the ranking of the articles is formed. Since the resulting list contains each article from the sample dataset without the cutoff, only IR-related metrics are used. Due to the volume of the annotated data, the results are not statistically tested.

The results in Table 1 show system performance for the query *What is known about Covid-19 transmission*. Reasonably high R-precision denotes that our model, as well as the baselines, is confident in the relevance of the highest-scoring articles. Furthermore, we assume that by utilizing the word and document level embeddings, the model has a chance to infer more subtle similarities between a query and an article which leads to notably higher precision at 15 and average precision. Traditional term-frequency-based approaches fail in such situations.

Similarly, the results for the query *What is known about Covid-19 incubation* are in Table 2. This time our model outperforms the baselines in the ranking of 5 highest-scoring articles. Even higher average precision than in previous case signals that the model, compared to the baselines, ranks relevant articles more consistently and closer to each other.

Table 1: Comparison of performance for the query *What is known about about Covid-19 transmission.*

Model	P@5	P@10	P@15	R-Prec	AveP
LM	0.80	0.60	0.47	0.67	0.69
BM25	0.80	0.70	0.47	<b>0.78</b>	0.70
Our Model	0.80	0.70	<b>0.60</b>	0.67	<b>0.84</b>

Table 2: Comparison of performance for the query *What is known about about Covid-19 incubation.*

Model	P@5	P@10	P@15	R-Prec	AveP
LM	0.60	0.50	0.33	0.60	0.64
BM25	0.40	0.50	0.33	0.40	0.55
Our Model	<b>0.80</b>	0.50	0.33	<b>0.80</b>	<b>0.97</b>

## 6. Conclusion

In this paper, we presented a novel task-specific model for the query-relevance ranking of the COVID-19 articles. The model combines a biomedical named entity recognition with word- and document-level embeddings to determine the relevance score of each article. We use a LM model and a BM25 ranking function as baselines, which our model outperforms on the sample dataset.

Note that our ultimate goal is to prove that a domain-specific model should be considered when tackling IR tasks.

As part of future work, we would like to increase the volume of the sample dataset. This would allow us to validate our results statistically. Additionally, when annotating articles we would like to use expert knowledge and greater coverage. Lastly, the comparison with deep learning approaches would be beneficial.

## References

- Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. 2020. SPECTER: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, Online, July. Association for Computational Linguistics.
- Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.
- Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei

Han. 2019. Cross-type biomedical named entity recognition with deep multi-task learning. *Bioinformatics*, 35(10):1745–1752.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214.

# Detection of Propaganda Techniques in News Articles

Karol Wilk

University of Zagreb, Faculty of Electrical Engineering and Computing  
Unska 3, 10000 Zagreb, Croatia  
{karol.wilk}@fer.com

## Abstract

When dealing with news articles one often encounters an underlying biased opinion in between the lines. While some authors include their opinion unintentionally in their articles there are others who deliberately try to convey their message subtly within the article. The latter case is by definition known as spreading propaganda. A variety of approaches in the field of Natural Language Processing (NLP) tasks have been already introduced to detect propaganda. Therefore, our aim is to evaluate very common transfer learning approaches used in the NLP Community and provide analysis of the results.

## 1. Introduction

Propaganda is primarily aimed at gaining the approval for a predefined agenda and influencing the opinions of the audience through psychological manipulation. Many mediums have had their share in spreading propaganda material. Classic mediums such as TV, radio and newspapers can be named as examples. The internet has contributed its share in transmitting such misinformation through social media and news articles. Due to its easy accessibility and numerous participants the propagandistic articles can reach the public in no time which make it very effective in achieving its goals. The means of these propagandistic articles can vary from influencing elections to manipulating the public opinion and having huge political impact (Lewandowsky et al. (2017)). Therefore, a big emphasize is to be put on identifying propaganda within news articles to consequently prevent the spread of propaganda. Following Da San Martino et al. (2019) there exist in total 18 different propaganda techniques. In table 1 the various techniques are listed.

In this paper, we compare a BiLSTM and the pre-trained BERT model on the propaganda detection task. We show the impact that these different approaches can make to propaganda detection and evaluate those approaches by their final classification score. As the baseline we choose a Random Forest Classifier. We evaluated the models according to their performance in solving the classification task in the SemEval 2020 Task 11. The SemEval 2020 Task 11 consists of the two subtasks Span Identification (SI) and Technique Classification (TC):

**SI** Given a sentence, predict whether it contains at least one propaganda technique.

**TC** Given a sentence, identify all spans of use of propaganda techniques in it and the type of technique.

In our work we focus on the SI task. This is a binary classification task and is evaluated based on the F1 score.

Table 1: List of Propaganda Techniques.

1	Presenting Irrelevant Data
2	Misrepresentation of Someone's Position
3	Whataboutism
4	Causal Oversimplification
5	Obfuscation, Intentional vagueness, Confusion
6	Appeal to authority
7	Black-and-white Fallacy, Dictatorship
8	Name calling or labeling
9	Loaded Language
10	Exaggeration or Minimisation
11	Flag-waving
12	Doubt
13	Appeal to fear/prejudice
14	Slogans
15	Thought-terminating cliché
16	Bandwagon
17	Reductio ad hitlerum
18	Repetition

## 2. Related Work

Before more attention to propaganda detection was paid there has been already done work on detecting "Fake News". This task was solved by similar approaches as it is done with propaganda detection. As fake news is also a type of propaganda where disinformation is intentionally spread through news outlets and/or social media it is not surprising that the same models which showed good performance on this task are now also applied to the propaganda detection task. A model which was used and performed good on this task is BERT.

Since BERT was first published in Devlin et al. (2018) it became state-of-the-art in various NLP tasks such as in Question Answering (Rajpurkar et al. (2018) and General Language Understanding Evaluation (Devlin et al. (2018)). BiLSTM was used as a part of a network in Wiedemann et al. (2018) where the network was used for a transfer learning approach to detect offensive language on twitter. There has been also already work done in the field of propaganda

detection. As in (Da San Martino et al. (2019) the BERT model has shown with 63.20% the best Precision of all models in the SLC (Sentencelevel Classification) task. In Roy et al. (2018) a BiLSTM was evaluated for a Fake News Detection task and scored with an accuracy of 42.65% only slightly lower than the proposed RNN-CNN model. A combination of BERT and BiLSTM was evaluated for the SLC task in Yoosuf and Yang (2019) and the BERT-BiLSTM outperformed in any metric a fine-tuned single BERT. In Pant et al. (2020) a BiLSTM and a BERT were compared as baselines to optimized and fine-tuned BERT-based models in a subjective bias detection task. While the BiLSTM was with a F1-Score of 64.8% clearly better than BERT, the BERT model was with an accuracy of 65.66% slightly better than the BiLSTM.

### 3. Dataset

The dataset consisted of roughly 446 news articles in .txt format. It was obtained from the SemEval 2020 Task 11: "Detection of Propaganda Techniques in News Articles" Competition. The news articles are split into a train set of 371 and a validation or test set of 75. This results roughly in a train/test ratio of 84/16. Moreover, the labels of the TC and SI task were separately given. The SI labels contained for each article the number of the lines where propaganda was given. Articles which didn't contain any propaganda were consequently left out in the list.

### 4. Approach

For the data loading, the pre-processing and the BiLSTM model the code provided by Bashir (2020) was used and formed as the backbone code of this project. Moreover, the outside library from Rajapakse (2020) was used for the BERT model.

The SI task is modelled as a binary classification problem. Given sequence of an article, the model should decide whether this sequence contains propaganda or not. As the news articles contained only words it was crucial to transform them to numerical vectors for the next steps. That is to make human language more transferable to a format more suitable for a machine to read. Therefore, the first step was to parse the news articles and convert each word to a number. After the sequences were transformed to a numerical representation, they were fed into the models. The test and training data were generally a two-dimensional array. The training data had for example a shape of (16673, 129).

To evaluate the models and compare them we used three metrics. These metrics are defined as follows:

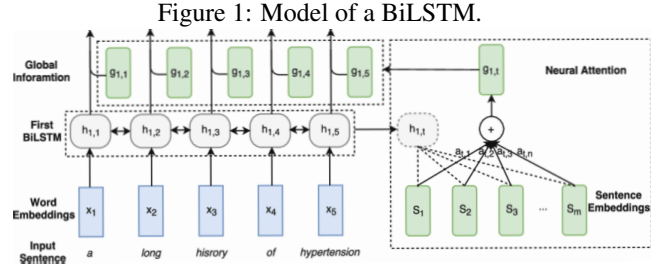
$$\text{Precision} = \frac{TP}{TP+FP} \quad (1)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (2)$$

where TP is the number of correctly predicted samples and FN is the number of samples that do belong to the right class, but were classified otherwise. And FP is the number of samples that don't belong to the class but were falsely classified as belonging to the class. The final F1 score is

obtained by calculating the harmonic mean of precision P and recall R:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$



## 5. Models

### 5.1. Baseline

For the baseline a Random Forest Classifier was chosen. As in (Aggarwal and Sadana (2019) a Random Forest Classifier reached a F1 score of 0.41 performing better than a SVM and only slightly worse than a Naive Bays model. Although the Naive Bays model performed better, it is considerable that strong dependencies in propaganda loaded articles exist. This means for example that if in an article a sequence with propagandistic content exists it is more likely that another sequence with propagandistic content exist in the article.

### 5.2. BiLSTM

As previously described, the BiLSTMs achieved good results in various text classification tasks. Bidirectional LSTMs learn from the input sequence from forward and reverse and therefore being generally faster and performing better than LSTMs. In fig. 1 a model of it can be seen. Pant et al. (2020) showed that a BiLSTM can also perform well despite competing with BERT. Because of that in this project a BiLSTM was also applied to the SI Task. The first layer of the BiLSTM was the embeddings that are used as input to a BiLSTM layer. The embedding size was 128. This gives a 128-dimensional representation of the input sequence. After that one BiLSTM Layer follows. To prevent overfitting a Dropout layer with a dropout rate of 0.5 was used. The output of the BiLSTM layer is then used as the input to the final fully connected layer with one neuron of which inputs correspond to being part of the class. For the network a sigmoid function was used because the sigmoid function is very often used for two-class classifications. Adam as the optimizer was used because of its generalization. It performs better than most of the optimizers.

The model was trained on 371 articles from the trainset for 10 epochs using the Mean Squared Error as the loss function and a batch size of 10. Evaluation was performed on the predicted class of each sample in the test set by the micro-averaged F1 score, the recall and the precision.

Table 2: Sentence Classification for different models.

Method	Recall	Precision	F1
Random Forest	44%	36%	40%
BiLSTM	49.7%	49.7%	45.4%
BERT	44.5%	61.4%	51.6%

### 5.3. BERT

The second model that was evaluated during the project was BERT. BERT was first introduced in Devlin et al. (2018) where it outperformed numerous models and since then become very famous. Currently, BERT-based models are the state-of-the-art in various text classification tasks. As there exist many different BERT models to find the most appropriate one is not easy. In contrast to the BiLSTM, the layer architecture was already constrained by the fixed BERT model provided by Hugging Face. The BERT model that was used was pre-trained on lower-cased English text and contained 12 layers, 12 sequence classification/regression heads and 110M parameters. It took an embedding size of 768 as input. BERT uses a specific input format because it consists of three embedding layers. BERT uses the WordPiece tokenizer to extract the token embedding, the segment embedding and the position embedding. These three embeddings are then summed up and fed into the Multi-head Self Attention layers which are in total 12 layers. After that, the model outputs the classification loss. Adam was chosen as the optimizer and a batch size of 32 was selected. The model was trained for 10 epochs and at the end the F1 score, the recall and the precision were evaluated.

## 6. Results and Analysis

In table 2 the rounded results are shown for the evaluation of the test set. It is clear to see that both, the BiLSTM and BERT, performed better than the baseline. Although, BERT performed worse in the recall than the BiLSTM, BERT performed with a F1 score of 51.6% at best in this project. The performance of BERT is not surprising as BERT is famous for outperforming other models and is therefore widely used as part of a transfer learning approach. Nonetheless, the BiLSTM also performed well and clearly was better than the baseline. It is also to mention that despite the BiLSTM having a very simple architecture compared to BERT it still scored the best results in recall. In conclusion, this paper showed that BERT’s state-of-the-art position is justified and also showed BERT’s ability to perform very well on the SI task of the SemEval 2020 Task 11 and clearly has no competition in this regard. For better results though BERT has to be trained on larger amount of training data and generally having more training capacities which involves the need of more computational power. Of course, the TC task can also be addressed as to further proof BERT’s performance. As in Yoosuf and Yang (2019) further works on propaganda detection can also include a combination of BERT and BiLSTM.

## References

- Kartik Aggarwal and Anubhav Sadana. 2019. NSIT@NLP4IF-2019: Propaganda detection from news articles using transfer learning. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 143–147, Hong Kong, China, November. Association for Computational Linguistics.
- Zahra Bashir. 2020. Task\_SI.ipynb [source code]. [https://github.com/zahrabashir98/Detection-of-Propaganda-Techniques/blob/master/Task\\_SI.ipynb](https://github.com/zahrabashir98/Detection-of-Propaganda-Techniques/blob/master/Task_SI.ipynb). [Online; accessed 28.06.2020].
- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news article. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China, November. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.
- Stephan Lewandowsky, Ullrich K.H. Ecker, and John Cook. 2017. Beyond misinformation: Understanding and coping with the “post-truth” era. *Journal of Applied Research in Memory and Cognition*, 6(4):353–369, December.
- Kartikey Pant, Tanvi Dadu, and Radhika Mamidi. 2020. Towards detection of subjective bias using contextualized word embeddings. In *Companion Proceedings of the Web Conference 2020, WWW ’20*, page 75–76, New York, NY, USA. Association for Computing Machinery.
- Thilina Rajapakse. 2020. simpletransformers. <https://github.com/ThilinaRajapakse/simpletransformers>. [Online; accessed 28.06.2020].
- Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don’t know: Unanswerable questions for squad. *CoRR*, abs/1806.03822.
- Arjun Roy, Kingshuk Basak, Asif Ekbal, and Pushpak Bhattacharyya. 2018. A deep ensemble framework for fake news detection and classification. *CoRR*, abs/1811.04670.
- Gregor Wiedemann, Eugen Ruppert, Raghav Jindal, and Chris Biemann. 2018. Transfer learning from LDA to

bilstm-cnn for offensive language detection in twitter.  
*CoRR*, abs/1811.02906.

Shehel Yoosuf and Yin Yang. 2019. Fine-grained propaganda detection with fine-tuned BERT. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 87–91, Hong Kong, China, November. Association for Computational Linguistics.

# Author Index

- Almer, Ivan, [1](#)  
Anić, Ivan, [5](#)
- Barišić, Luka, [9](#)  
Belić, Bruno, [13](#)  
Bertović, Matija, [17](#)  
Bilić, Ivan, [5](#)  
Bogdanović, Vedran, [22](#)  
Buhiniček, Nikola, [60](#)
- Čaušević, Adi, [22](#)  
Ćavar, Katarina, [32](#)  
Čović, Nikolina, [27](#)  
Cvitković, Marko, [27](#)
- Deur, Sanja, [32](#)  
Dorić, Gabriela, [37](#)  
Duspara, Ivica, [37](#)
- Fernandez, Luis, [46](#)
- Gigot, Florian, [42](#)  
Goluža, Sven, [1](#)  
Grgurina, Simon, [13](#)
- Jagorel, Gaétan, [42](#)  
Jurčić, Krunoslav, [51](#)
- Kršić, Marko, [37](#)
- Lovrenčić, Ivan, [60](#)  
Lugar, Barbara, [13](#)
- Magdić, Antun, [17](#)  
Marić, Patrik, [27](#)  
Marsault, Barthélémy, [42](#)  
Mustapić, Tina, [1](#)
- Paulsen, Markus, [46](#)  
Pavlović, Luka, [51](#)  
Pisačić, Marko, [9](#)  
Protrka, Dorotea, [32](#)  
Pušelj, Dora, [55](#)
- Radović, Filip, [9](#)
- Šaško, Mario, [60](#)  
Škalec, Tena, [55](#)  
Škudar, Silvije, [5](#)  
Sršen, Martin, [51](#)
- Vladika, Juraj, [22](#)
- Wang, Bingji, [46](#)  
Wilk, Karol, [64](#)
- Žužul, Ante, [17](#)