UNIVERSITY OF TWENTE.
Natural Language Processing

Exploring different n-gram models and embedding techniques in automatic fake news detection

Alessandro Cortese (id. 3105938)
Gianmarco Lodi (id. 3103544)
Project Group 16

31 October 2022

1 Introduction

The pervasiveness of social media in everyday life has made sharing news an extremely quick and straightforward activity. If, on one hand, this can lead to more informed and engaged citizens, on the other hand, social media platforms represent a fertile ground for the spread of fake news. This can be defined as "false stories that appear to be news, spread on the internet or using other media, usually created to influence political views or as a joke"[1].

Several studies noted that the diffusion of misleading information has harmful effects on society and individuals. For example, Van Duyn & Collier (2019) argue that fake news undermines public trust in the media as a whole [1], whereas Persily (2017) claims that fake news is currently threatening the democratic process as it "renders it less likely that voters will choose on the basis of genuine information rather than lies or misleading *spin*" [2]. Moreover, the recent Covid-19 pandemic caused a surge in the proliferation of conspiracy theories and discredited science, leading as far as promoting dangerous health advice and potentially life-threatening behaviours [3].

As a result, the automatic detection of fake news has lately emerged as an area of research that is receiving a great amount of interest for its extremely useful implications in lessening the detrimental effects of fake news on society.

The aim of this project is to investigate how different Natural Language Processing models and techniques affect the performance of two Machine Learning classifiers trained to predict if a political statement is true or false. Furthermore, we would like to understand whether the textual features alone are sufficient to achieve good results in this task, and if embedding the statements' metadata in the models can improve their performance.

2 Related Work

There are several methods which have been thus far employed in automatic fake news detection. In particular, Shu *et al.* (2017) list five approaches to extract features for this task [4]:

- the *linguistic-based*, which uses linguistic features that capture the different writing styles and sensational headlines typical of fake news;
- the *visual-based*, which retrieves features from visual elements (e.g. images or videos);
- the *user-based*, which exploits the characteristics of the users interacting with the news on social media;
- the *post-based*, which focuses on identifying useful information via the reactions and comments expressed on social media posts;
- the *network-based*, whereby features are extracted by constructing specific networks among users who published related posts.

In this work, we are focusing on the *linguistic-based* approach, thus identifying fake news by considering textual features. This is based on the assumption that "linguistic behaviors such as punctuation usage, word type choices, part-of-speech tags, and emotional valence of a text are rather involuntary and therefore outside of the author's control, thus revealing important insights into the nature of the text" (Perez-Rosas *et al.*, 2017) [5].

This specific methodology has been widely employed for similar supervised fake news detection tasks in the past. Oshikawa *et al.* (2018) collected textual features from statements, posts, tweets and articles found in multiple datasets and compared experimental results of different methods,

[1]Source: Cambridge Advanced Learner's Dictionary & Thesaurus (`https://dictionary.cambridge.org/dictionary/english/fake-news`)

focusing on Neural Network models [6]. Granik and Mesyura (2017) [7] have applied a binary Naive Bayes classifier on Facebook posts, whereas Singhal *et al.* (2019) have combined BERT as a feature extractor for tweets and VGG-16 as a feature extractor for their associated images, obtaining promising results [8].

Yet, these works place much focus on the Machine Learning tasks themselves, whereas our main goal is to delve deeper into exploring different embedding techniques, varying their configurations and trying to see which are the most suitable for our task. Moreover, simple bag-of-word approaches are often overlooked in these studies, but we would like to explore them nonetheless, as they might lead to interesting insight.

3 Data

To conduct our analysis we used the LIAR[2] dataset, which contains 12824 short statements manually labeled by the fact-checking website PolitiFact[3] according to their degree of truthfulness (i.e. “pants-on-fire”, “false”, “barely true”, “half-true”, “mostly true”, “true”). The statements are 2 to 73 words long, with an average length of 18 words [9].

Each statement also comes with a series of metadata, namely: the state where the sentence was pronounced, the topics of the sentence, the particular occasion in which it was said, the speaker’s name, their job, their party affiliation and their truth credit history. This last piece of data consists of the total counts of the various sentences pronounced by that specific speaker by their degree of truthfulness. However, even though the labels are six, the columns related to the credit history are five: the missing one corresponds to the “true” statements count.

In Table 1 we report an example of a record in the LIAR dataset.

4 Methods

4.1 Data preprocessing

The dataset came already divided in train ($n = 10268$), validation ($n = 1284$) and testing ($n = 1272$), so we used this precomputed split.

One thing that can be immediately noticed is that there are some rows which contain formatting problems. For instance, the sentence whose id is “1606.json” contains actually other ten observations inside which get recognized as a single one, because of a syntax problem in the .tsv file. We manually corrected such issues, as there were just a few instances of them.

Moreover, some of the sentences in the LIAR dataset are clearly mislabeled. For example, the sentence “5919.json” consists of just “On abortion”. This is because it comes from the wrong set of data (PolitiFact’s Flip-o-Meter rather than its Truth-o-Meter), and consists of the title of an article about Mitt Romney’s significant shift in his position on abortion. However, the sentence is still tagged as “false” and as a result such kind of datapoints are not useful for our analyses. We filtered out these observations by discarding every sentence whose length in words is less than 8 and starts with “On ”, as all these kind of observations presented this common pattern. However, we do not exclude the presence of other longer titles which may have not been filtered out.

As far as the labels are concerned, the in-between assertions marked as “half-true” and “barely-true” have been eliminated, as our focus consists in differentiating between the extreme cases of falsehood and truthfulness. This reduced the dataset to 7993 sentences. Then, the labels “true” and “mostly true” were combined into one single “True” category (coded as 1), while the labels

[2] `https://cs.ucsb.edu/~william/data/liar_dataset.zip`
[3] `https://politifact.com`

id	label	statement	subjects	speaker	speaker_job_title	state_info	party_affiliation	count_1	count_2	count_3	count_4	count_5	context
3652.json	false	President Barack Obama took exactly none of his own deficit reduction commissions ideas. Not one.	deficit, federal-budget	John Boehner	Speaker of the House of Representatives	Ohio	republican	13	22	11	4	2	an interview with ABC news

Table 1: Example of an observation in the LIAR dataset before preprocessing.

id	label	statement	subjects	state_info	party_affiliation	context
3652.json	0	presid barack obama deficit reduct commiss idea	deficit, federal-budget	Ohio	republican	an interview with ABC news

Table 2: Example in Table 1 after preprocessing.

"false" and "pants-on-fire" were placed in the broader "False" category (coded as 0). This resulted in 4508 sentences having class 1 and 3485 sentences having class 0.

We then tokenized the sentences, applied case folding, removed stopwords and punctuation, and applied Porter stemming. All these operations were done using the NLTK python library.

As far as the metadata are concerned, the ones we considered are:

- `subject`: there are 142 unique subjects. Each statement can have more than one subject.
- `state_info` : the variable contains 56 unique values.
- `party_affiliation`: since the rows corresponding to a republican or democrat speaker are the vast majority, all the remaining values are renamed as "other". However, "none" values are not deleted as expression of non-identification of the speaker with any party. Therefore, there are 4 different unique values.
- `context`: there are 3648 unique values for this variable.

The other metadata (i.e. `speaker`, `speaker_job_title` and the five columns related to the credit history) have been discarded, as the first two contain too many unique values and NAs, whereas the credit history does not contain the number of true statements count, thus skewing the data towards the false and pants-on-fire statements.

In Table 2 we show the same example of Table 1 after this preprocessing procedure.

4.2 Analysis

We used three different techniques to turn the statements into feature vectors:

1. A bag-of-words model with *tf-idf* weighting.
2. A *Word2Vec* model.
3. A *Doc2Vec* model.

For the first two models we also tested several configurations. In particular:

- For the first model, we tried using unigrams, bigrams, trigrams and a mixed model employing both unigrams and bigrams, by changing the `ngram_range` parameter in scikit-learn `TfidfVectorizer`. In order to keep the dimensionality of the *tf-idf* document-term matrices manageable, the `min_df` parameter was set to 3, thus ignoring each n-gram whose frequency in the vocabulary was less than that.

- For the second model, we employed unigrams as well as multi-word expressions, using gensim `models.phrases`. We computed the vector for each sentence by averaging the words vectors, using both a simple and a *tf-idf* weighted average. We also explored a model pre-trained on a large corpus, such as the Google News dataset. We also tested different values for the *Word2Vec* class parameters: we experimented with different vector sizes (`vector_size = 50, 100, 200, 300`), window sizes (`window = 2, 5, 10`) and thresholds for the minimum count value (`min_count = 1, 3, 5`); we also explored whether applying *CBOW* algorithm or *skip-gram* (`sg = 0, 1`), and whether using negative sampling or not, specifying the number of "noise words" (`negative = 0, 10, 20`). Since the results of these different configurations were extremely similar, we decided to keep the default parameters.

The embeddings derived by each of these models were used as input for two binary classifiers, namely a Support Vector Machine (SVM) classifier with linear kernel and a Random Forest (RF) model, as implemented by scikit-learn.

A grid search was performed to tune the `cost` hyperparameter for the SVM and the `n_estimators` hyperparameter for the RF. The best model was chosen according to the accuracy obtained on the validation set and was used to predict the values on the test set. We then registered the accuracy of the classifier on the test data and plotted the resulting confusion matrix.

Additionally, for the *tf-idf* models, the 15 most important features for both classifiers were calculated and plotted. For the SVM classifier, feature importance was obtained by looking at the largest coefficients, whereas for the RF classifier it was based on mean decrease in impurity, namely the total decrease in node impurity, weighted by the probability of reaching that node and averaged over all trees of the ensemble.

We then tried to answer our second research question by including the sentences' metadata as features for the two models with the highest test accuracy. For this purpose, `subject`, `state_info` and `party_affiliation` were one-hot encoded. As far as `context` is concerned, since a great deal of its values contain similar information (e.g. "a TV ad", "a television ad", "a TV advertising" are all different contexts) we concatenated each context string to its statement, and then preprocessed them as described in 4.1. We then repeated the classification task using also these additional features.

5 Results

As it can be noticed from Table 3, the models without metadata perform rather similarly, as their accuracy is around 57–63%. The choice of the classifier does not impact the results much, as well. Also, it can be seen that simpler models, such as the ones based on *tf-idf*, perform as well as the more sophisticated word embeddings models, if not even better. The best model with *tf-idf* embeddings is the one which mixed unigrams and bigrams, whereas the best one with respect to the *Word2Vec* embeddings is one pre-trained on Google News. Moreover, the former is also the model which leads to the highest test accuracy using the Random Forest classifier, whereas the latter has the highest test accuracy with the SVM.

It can also be observed that bigrams and trigrams perform worse than unigrams. We argue that this is due to the relatively small size of the dataset, which implies low count frequencies using higher-order n-grams, thus leading to sparse models which do not generalize well to unseen data.

Tf-idf Model	SVM	RF
Unigrams	0.612	0.616
Bigrams	0.604	0.587
Trigrams	0.593	0.585
Mixed Uni-Bi	**0.624**	**0.628**

Word2Vec Model	SVM	RF
Simple Avg	0.585	0.590
Weighted Avg	0.575	0.575
Bigrams Simple Avg	0.578	0.576
Google News Simple Avg	**0.631**	**0.602**

Model	SVM	RF
Doc2Vec	0.592	0.596

Table 3: Accuracy scores for each model, without metadata.

Model	SVM	RF
Tf-idf Mixed	0.649	0.668
W2V Google News	0.677	0.658

Table 4: Accuracy scores for the two best models in Table 3 with metadata.

Regarding the performances of the *Doc2Vec* model, which directly creates vector embeddings for a document or a sentence, the classifiers could not achieve higher than 60% in accuracy.

Adding metadata brings a gain of ∼2.5–4.5% in the accuracy scores (Table 4) of the two models we considered. In fact, the foremost performance is obtained with the pre-trained word embeddings on Google News, averaged for each statement concatenated with the context string, merged with metadata features and trained on the SVM classifier (accuracy = 67.7%).

Looking at the confusion matrices, we noticed that usually the number of False Positive is greater than that of False Negatives by 50–100 units. This could be due to the slight class imbalance. Moreover, on the *Word2Vec* bigram model, the SVM classifier predicted everything as belonging to the majority class (i.e. class 1). The feature importance plot and the confusion matrix in Figure 1 are obtained from the RF classifier using *tf-idf* mixed model with metadata, which is the second-best one. For this specific model, it looks like the democrat or republican affiliation of the speaker are the two most important features, followed by the words *interview* (which we assume coming from the context), *percent* and *obama*. We did not compute the feature importance of the *Word2Vec* and *Doc2Vec* models, as their features are not interpretable.

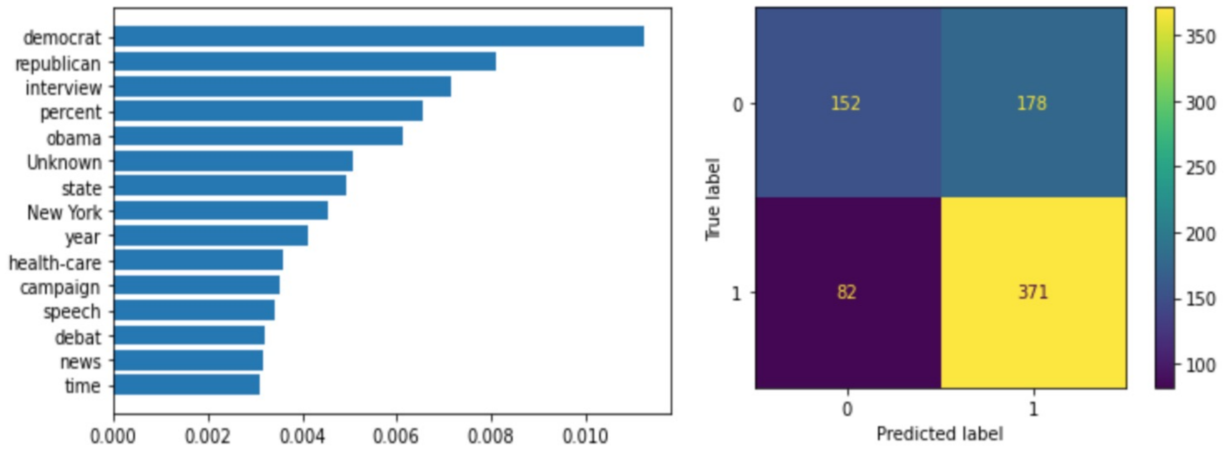

Figure 1: Random Forest feature importance plot and confusion matrix for the *tf-idf* mixed model with metadata.

6 Discussion

6.1 Error analysis

We investigated the errors made by our system looking at the misclassified observations for the second-best model with the RF classifier, exploring the impact of the most important features and taking into account its confusion matrix for the predictions, as shown in Figure 1. It can be observed that, on the test set, 73% of the sentences pronounced by democrats belong to class 1. However, 92% of the democrats' statements are predicted as class 1. This means that our model overestimates the probability of belonging to class 1 if the speaker is a democrat. Indeed, 84% of misclassified democrats' sentences are False Positives.

Another interesting feature is the word *obama*. On the test set, 47% of the sentences having this word belong to class 0 and 53% to class 1. However, just 29% of these sentences are predicted to class 1 by our model. 83% of the misclassified sentences in which the speaker used the word *obama* are False Negatives. Therefore, our hypothesis is that in our corpus there are many sentences containing lies regarding Barack Obama.

6.2 Limitations and further work

One reason that could explain the poor performances of *Word2Vec* and *Doc2Vec* is the relatively small size of the dataset, which makes it difficult for these models to extract meaningful embeddings. This would also explain why the *Word2Vec* model trained on the large Google News corpus (100 billion words) has the highest accuracy. Another issue could come from the fact that, even though we performed extensive data cleaning, it is not possible to exclude the presence of mislabeled observations which hamper the models' training.

More limitations arise by the fact that we did not apply cross-validation nor performed extensive hyperparameter tuning, because of computational reasons. These are two important steps in the Machine Learning workflow and, if we had to start over, we would focus more on these aspects.

Additionally, the specific models we used are much simpler than the most state-of-the art methods. Indeed, it would be interesting to investigate whether more sophisticated Natural Language Processing tools, such as transformers-based methods like BERT, could lead to higher classification accuracy.

Overall, it appears that our approach is clearly too simple for the task at hand. For example, Reis *et al.* (2019), when extracting textual features for a similar supervised fake news classification task, make use of a variety of psycholinguistic features from LIWC to extract signals of persuasive and biased language, as well as part-of-speech counts, punctuation usage and various other semantic features, which we have not considered. [10] [11]

However, we argue that this work has highlighted certain underlying problems with automatic fake news detection.

Firstly, as reported by Shu *et al.* (2017), there are many other non-textual features which can be extracted from news. For example, with respect to political statements, facial expressions, gestures and tone of voice of the speaker could be extremely useful to detect deceit.

Moreover, it is well documented that identifying irony is a difficult task for NLP systems (Wallace, 2015): this represents an issue when fake news are purposefully created as jokes. [12]

Finally, we used a binary model but reality is much more nuanced than that. The problem with fake news is that most of the time they are not "pants-on-fire" false, but their deceit rather lies in the details, or in the way the information is conveyed. In this regard, we argue that it would be more useful to develop a model which assigns a specific "truthfulness" score in a continuous range, instead of a binary classifier. However, we recognize that since most of the accessible datasets have discrete ground truth labels, it would be challenging to translate these into numerical scores (Oshikawa *et al.*, 2018) [6].

References

[1] Emily Van Duyn and Jessica Collier. Priming and fake news: The effects of elite discourse on evaluations of news media. *Mass Communication and Society*, 22(1):29–48, 2019.

[2] Nathaniel Persily. The 2016 us election: Can democracy survive the internet? *Journal of democracy*, 28(2):63–76, 2017.

[3] Sander van Der Linden, Jon Roozenbeek, and Josh Compton. Inoculating against fake news about covid-19. *Frontiers in psychology*, 11:566790, 2020.

[4] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1):22–36, 2017.

[5] Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*, 2017.

[6] Ray Oshikawa, Jing Qian, and William Yang Wang. A survey on natural language processing for fake news detection. *arXiv preprint arXiv:1811.00770*, 2018.

[7] Mykhailo Granik and Volodymyr Mesyura. Fake news detection using naive bayes classifier. In *2017 IEEE first Ukraine conference on electrical and computer engineering (UKRCON)*, pages 900–903. IEEE, 2017.

[8] Shivangi Singhal, Rajiv Ratn Shah, Tanmoy Chakraborty, Ponnurangam Kumaraguru, and Shin'ichi Satoh. Spotfake: A multi-modal framework for fake news detection. In *2019 IEEE fifth international conference on multimedia big data (BigMM)*, pages 39–47. IEEE, 2019.

[9] William Yang Wang. Liar, liar pants on fire: A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017.

[10] James W Pennebaker, Martha E Francis, and Roger J Booth. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001, 2001.

[11] Julio CS Reis, André Correia, Fabrício Murai, Adriano Veloso, and Fabrício Benevenuto. Supervised learning for fake news detection. *IEEE Intelligent Systems*, 34(2):76–81, 2019.

[12] Byron C Wallace. Computational irony: A survey and new perspectives. *Artificial intelligence review*, 43(4):467–483, 2015.