Profiling Fake News Spreaders on Twitter
Notebook for PAN at CLEF 2020

Álvaro López and Pasqual Martí

Universitat Politècnica de València
allochi@prhlt.upv.es, pasmargi@inf.upv.es

Abstract. The increase of fake news makes social media a dangerous and powerful tool due to the big impact that it can have on society. Fake news must be detected automatically to avoid the mass manipulation of people. In this paper we present our team participation at PAN 2020 Shared Task: Profiling Fake News Spreaders on Twitter. We propose a deep learning model to classify authors from twitter into fake news spreaders or not according to their their tweets. The main problem that we tackle is the classification for Spanish and English authors separately, although we also considered bilingual models to solve the task. Our best model obtained an accuracy of 0.755 on Spanish and 0.68 on English.

1 Introduction

Global access to the internet has communicated people from every part of the world and every social background. When it first appeared, the Internet was intended to be an independent media, free from the pressure of governments and big influential companies. Its users found in social media platforms a door to all sorts of free information. However, because of its reachability, the Internet is also full of false, intentionally or unintentionally misleading, information. Malicious agents take advantage of social media to spread fake news. Fake news capitalize on society's polarization, appealing to the user's feelings, which makes them be easily shared without questioning their credibility. This makes fakes news a big threat to our society and a powerful tool to manipulate people, ultimately having a great impact on their lives.

Currently, one of the most popular social media is Twitter, a platform in which users can share tweets, short written messages that may contain pictures, videos or links to other websites. In this work, we are presented with the task of classifying Twitter users in two classes depending on whether they are fake news spreaders or not. The aim is to build a system that quickly detects this type of user before the news are spread or, at least, stops them from spreading more.

In this paper, we present our participation at PAN 2020 Shared Task: Profiling Fake News Spreaders on Twitter (4). Our approach is based on deep learning

classification models for every tweet language as well as a bilingual model that classifies tweets of both languages.

2 Related work

The detection of fake news is a complex topic which can be addressed in very different ways in order to make the most of the available information. In (6), authors use data mining techniques to make use of all possible information before the classification. They use not only a linguistic-based approach but also get data from the author profile, their network of contacts in social media (to detect bots) and other user's opinions in their posts.

The type of model used for the classification varies too. In (1) they used a Naive Bayes classifier to tackle the task in a similar way that spam filters do. On the other hand, in (2) authors used SVM classifiers with linguistic features such as n-grams, punctuation, psycholinguistic features (aided by a lexicon), readability and syntax. Finally, in (5) authors use a deep learning model that combined linguistic information extracted from news using an RNN and extracted information from user features with an FC network, building an ensemble of models to classify.

We have observed a diversity in approaches which might be derived from the heterogeneity among fake news sources. The media from where the data is extracted greatly influences the news extension and format, since the information is presented differently in a tweet than in a web post, for instance. Besides that, the metadata regarding the author and her/his social media network differs too. Because of that, there is not just one predominant approach, although it must be noted that most of them take into account linguistic features.

3 Our approach

In order to classify the authors we are given some of their tweets, all written in either English or Spanish. We approached the problem in two different ways: on the one hand, training a classifier that takes all tweets as a sequence and assigns a class to the author; on the other hand, training a single tweet classifier and then performing a vote with the classification of each tweet to decide the class of the author.

Between both approaches, we decided to use the single tweet classifier for two main reasons: firstly, we prefer to train a solid single tweet classifier so that we do not depend on having more or fewer tweets than in inference, and we will be able to classify correctly new authors even if they have fewer or higher number of tweets than the ones in our training set. Secondly, if we use the approach of the sequence of tweets for each author, the number of samples to train the models is reduced drastically since the number of single tweets is significantly higher than the number of authors.

In the next sections, we are going to describe the models implemented following the chosen approach and how their training was performed. We will present

not only independent models for English and Spanish but also a bilingual model that deals with the problem of classifying authors whose language is unknown as well as bilingual authors.

4 Models

In this section we describe the model architectures employed and the main idea behind them. The models used can be grouped in three classes based on the type of encoding used for the tweets: untrained embedding, pretrained neural net language model and pretrained multilingual encoder.

The first type is **model 1** which uses a new untrained embedding layer to encode the tweets words. After this embedding layer it has two bidirectional LSTM layers with 32 units and then a fully connected part with one dense layer of 128 neurons and the last one with 1 neuron with sigmoid activation function for classification (0 not fake, 1 fake). For the other layers we employed the ReLU activation function and batch normalization between all the layers. To avoid overfitting we used a dropout layer between the dense layers with a drop factor of 0.5. With this model we tried to train our own embeddings for English and Spanish from scratch. But as we show in the results section we couldn't achieve good results because we did not have enough data to train the embeddings and, consequently, the models with this new embeddings couldn't learn well because of the bad encoding of the words.

The second type of models are the ones with a pretrained embedding based on feed-forward Neural Net Languange Model (nnlm)(8)[1], the size of the embedding output is a vector of 128 dimensions with the full tweet encoding. This is because this module internally combines the words embeddings into a single vetor to encode the sentence. With this pretrained embedding we aimed to improve the results taking advantage of the better representation of the tweets that it gives thanks to the large amount of data[2] and resources used to train it.

We trained two models with this kind of embedding. The first one is **model 3**, this model is a basic fully connected net with two dense layers of 32 neurons and the last one with 1 for making the classification with a sigmoid activation function. Before these three layers we employed batch normalization and a drop out with a drop factor of 0.5 to avoid overfitting and to regularize the training of the parameters. With this regularization techniques we tried to avoid destroying our embedding since it had been fine tuned during the training phase.

The second model trained with the nnlm based embedding is **model 4**. This one uses a slightly more complex architecture as can be seen in Figure 1. It also uses dense layers but we introduce skip connections that allow to jump over each dense layer to give more paths to back-propagate the error. To implement the skip connections we used concatenation layers, and to apply

[1] We get this nnlm models from tensorflow hub (`https://tfhub.dev/google/collections/tf2-preview-nnlm/1`).

[2] The datasets used to train these embeddings are the English Google News 200B and the Spanish Google News 50B.

regularization we used batch normalization after each dense layer and dropout after each concatenation layer with a drop rate of 0.5.

Finally the last type of models are the ones with a pretrained multilingual encoder[3]. This encoder is a CNN based model trained over 16 languages for tasks such as text classification, text clustering, semantic textural similarity retrieval, cross-lingual text retrieval, etc. All the model details can be found in (7). The main idea of this model is to be able to encode sentences in different languages but with the same meaning to vectors with a similar encoding. In this case the encoder generates a 512 size vector to encode the whole input sentence. By using this type of sentence encoder, not only we have been able to train models for English and Spanish but also for both languages at the same time, aiming to get similar results to the single language models.

So based in this type of sentence encoder, not only we have been able to train models for English and Spanish but also for both at the same time trying to get similar results with this single model.

We trained three models with the multilingual encoder. The first one is **model 5**. This model shares exactly the same architecture than model 4 (shown in Figure 1) but changing the nnlm encoder by the cnn multilingual encoder. We used the same architecture to be able to compare directly the two encoders and because the capacity of this model is big enough also for this encoding.

The next model is **model 6** which is an extension of model 5, since it has the same layers but with double number of neurons in the fully-connected layers. With this modification we tried to learn more patterns in the data thanks to the bigger model and the strong regularization of the dropout layers to force sparsity.

The last model trained is the bilingual one, **model 7**. To approach this harder problem of multilingual classification we take advantage of the multilingual encoder to be able to work with nearly the same representation of the tweet no matter the language it is written in. But as we are training for a bilingual task, in order to force the network to learn some finer characteristics of each language we trained a bilinear model with two parallel networks that have been pretrained each one for a single language. For that, we used two pretrained versions of model 5, one for English and one for Spanish, and we connected the output of the cnn encoding to both nets; the resulting output tensors of these nets are then concatenated and passed through a simple fully-connected net with a dense layer with 128 neurons and ReLu, and the output dense with 1 neuron and sigmoid activation function. We also used batch normalization before the two dense layers.

[3] We get this multilingual encoder from tensorflow hub (`https://tfhub.dev/google/universal-sentence-encoder-multilingual-large/3`).

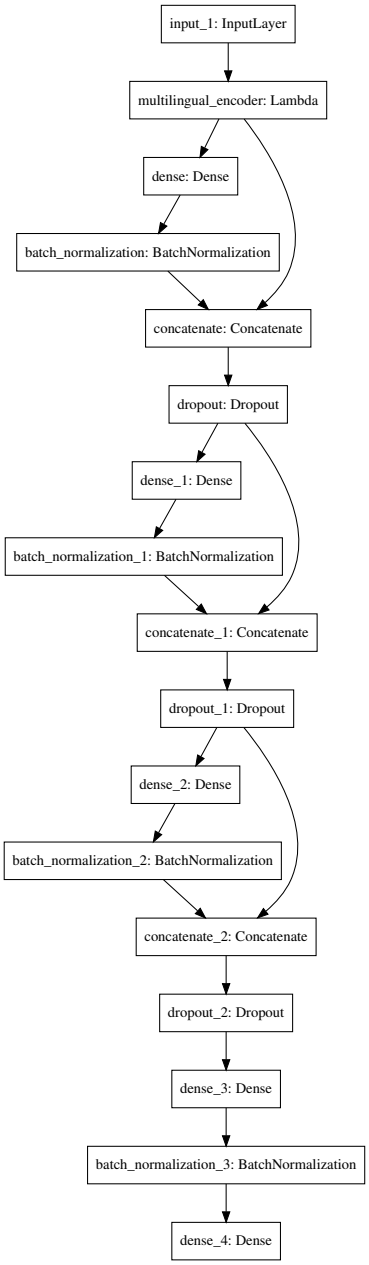


Fig. 1: Model 4 architecture. The dense layers units are 32, 64, 128, 256 and 1 (from top to bottom). The drop rate for the dropout is 0.5. The activation functions are ReLU except for the last dense which is sigmoid.

5 Experimental setup

5.1 Data processing and partitioning

In this section we describe how we split and processed the data in order to train the models.

The dataset has tweets from 600 authors written in English or Spanish. For this task we will know in advance the language of the tweets of an author, so we will be able to build independent models for each language. Nevertheless, we will also confront the task with a multilingual model that works with tweets of both languages.

As can be seen in Table 1 we have a perfectly balanced dataset at author level so we decided to make a random partition with 80% of the authors for train split and the other 20% for validation. Since the models were tested on the TIRA(3) platform, we avoided creating a test split so as to have more training data.

From this partitions we extracted all the tweets and their label from the author and stored them in one file for each partition, obtaining a dataset to train the single tweet classifiers. All these tweets have been processed to be in one single line by removing the break line token ('\n').

After this we obtained the data balance shown in Tables 2 for English and 3 for Spanish.

Language	**Class 1**	**Class 0**	**Total**
English	150	150	300
Spanish	150	150	300
	300	300	600

Table 1: Statistics of the dataset authors by label and language.

Partition	**Class 1**	**Class 0**	**Total**
Train	12400 (52%)	11600 (48%)	24000
Validation	2600 (43%)	3400 (57%)	6000
	15000	15000	30000

Table 2: Statistics of the dataset tweets by label for English data. The percentages shown are the percentage over the split of data (train or validation).

We have implemented two input pipelines because not all the models previously described accept the same input data format.

The first pipeline, used for model 1, takes the tweets strings and tokenizes them separating the tokens by the blank space and then uses a token encoder to get the id corresponding to each word token. We used one token encoder with the English vocabulary and another for Spanish, both of them taking the vocabulary set of words from the tweets in the train partition.

Partition	Class 1	Class 0	Total
Train	11800 (49%)	12200 (51%)	24000
Validation	3200 (53%)	2800 (47%)	6000
	15000	15000	30000

Table 3: Statistics of the dataset tweets by label for Spanish data. The percentages shown are the percentage over the split of data (train or validation).

For the rest of the models we used the other input pipeline that just forwards the tweets strings to the model. This is because the pretrained encoders (nnlm and multilingual) have their own sentence processing from the sentence string to the feature vector.

5.2 Training

As we explained in the description of the models, we applied abundant regularization with dropout and batch normalization. This is because we have observed that without it the models quickly overfitted reaching training accuracies over 90% while having validation accuracies just over 60%. With all the regularization we achieved a very controlled training with very low overfitting and in very advanced epochs.

Regarding models 1, 3 and 4, the ones with the new embedding and the nnlm encoder, we trained them for 5000 epochs, which was possible thanks to their training speed of two seconds per epoch during training. Once again, to avoid overfitting and develop a more stable training we used the SGD optimizer with a momentum of 0.9 and a low learning rate of 0.0002, which is lowered by a learning rate scheduler to 0.0001 from epoch 1000 and to 0.00005 from epoch 3000.

For the models with the cnn based multilingual encoder we had to adjust the number of epochs because they required more computation than the previous ones. In this case we trained for 1000 epochs using the Adam optimizer with a learning rate of 0.0002.

6 Results

For getting the vote result for classifying an author we implemented two types of voting techniques given the probability of each tweet of being fake. These two voting methods are the averaged vote and product vote.

The averaged vote is a more standard way of performing the voting process. The author will be classified as a likely fake news spreader depending on if the average of the probabilities of their tweets is higher than a given threshold.

The product vote is slightly more sophisticated and consists of taking again the probabilities of each tweet of being fake given by the model and we also compute the probability of being true for each tweet ($P(true) = 1 - P(fake)$). Then we perform the product of of each group of probabilities and we get the

author label by choosing the class of the highest product result. This method is shown in equation 1 where m is the total number of tweets for this author. Note that in this case we don't use any threshold to get the label, we just take the class that maximizes the product. With this vote technique we aim to get better results in the case of having very high probabilities because they have less impact to reduce the product value.

$$author\_pred = argmax(\prod_{i=0}^{m-1} P(tweet_i = true), \prod_{i=0}^{m-1} P(tweet_i = fake)) \quad (1)$$

After making inference in the validation set with the best model of each type we got the results of Table 4. Note that for all the results shown we obtained the same value with both voting techniques described except in two cases (marked with *) where we got modestly better results using the product vote. The table also shows the false negatives (FN) and the false positives (FP) in order to make a better analysis.

	English		**Spanish**		**Bilingual**	
Model	*Accuracy*	*FN/FP*	*Accuracy*	*FN/FP*	*Accuracy*	*FN/FP*
1	0.57	26/0	0.47	32/0		
3	0.58	8/17	0.75	9/6		
4	0.67	9/11	0.75*	10/5		
5	0.73	9/7	0.73	11/5	0.68*	18/21
6	0.73	13/3	0.73	9/7	0.66	24/17
7					0.66	27/14

Table 4: Results on the validation split for every model and language. In the case of bilingual results the accuracy is computed over the combination of the English and Spanish validation splits. *In this case the product voting obtains better results.

Analyzing **model 1** results, if we look at the accuracies obtained and the FN/FP ratio we can see that this model is just classifying always in the "true" class because the obtained accuracy matches the number of true samples in the validation sets for English and Spanish (see Tables 2 and 3). This is because of the model embedding which is not pretrained and could not be trained with the available data and resources. If the embedding is bad the results of the net will also be bad.

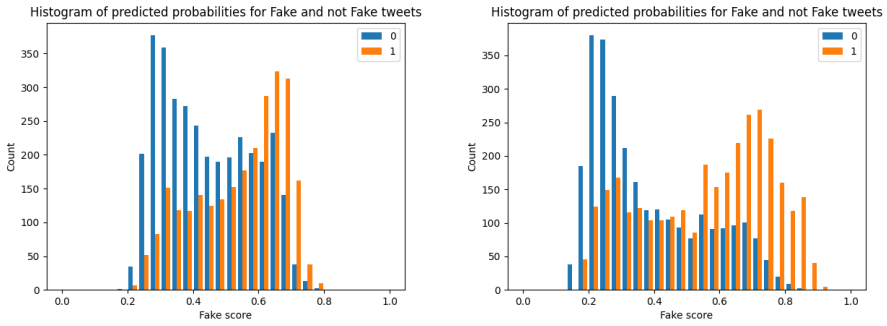
Looking at **models 3 and 4** we can see how we started to get some better results, mainly in the case of the Spanish data that seems to be an easier task than the English one. In the case of model 3 and English data we get just one more point in accuracy over the partition proportion but looking at the FN/FP we see that the main source of errors are the FP so this model is not voting just the majority class. However, a big step is achieved by model 4, which has a more

complex model architecture and gets significantly better results in the case of English data.

Finally looking at **models 5, 6 and 7**, the ones with the multilingual encoder, we see a huge increase in the case of English data and a smaller decrease for the Spanish, but in overall the performance of the models 6 and 7 get a better result for classifying both languages separately. Note that model 6 is just a bigger version of model 5 but it doesn't achieve any better results and, looking at the bilingual classification, model 5 gets better results with the product vote. Also note that model 7 gets the same result as model 6 despite having the pretrained models 5 and 6 as two branches of its net.

We also analyzed the models output probabilities to see if the predictions are made with a good level of confidence by the models. In Figure 2 we show the counts of probabilities outputted by model 5 making inference with the development partition in both languages. Comparing both languages we can confirm that for Spanish the models obtained better results because the fake samples probabilities are more separated from the true ones in the right histogram. Also, the range of probabilities is wider in Spanish so it has a more confident prediction in some cases.

In the case of having more data to make this histogram analysis we could adjust the threshold for making the average vote behave in a way such that the number of false negatives is minimized, lowering the threshold from 0.5 and making a trade-off between false positives and false negatives, taking into account the shape of the histogram. We made some experiments by changing the threshold and we got better results in some cases but since the development partition is very small and we can not ensure that it is a representative set of data, we decided to not implement it in the final model.



(a) English (b) Spanish

Fig. 2: Probability count for the predictions made for each class and dataset language.

7 Test set submission on TIRA

After analyzing the results we decided to select model 5 as the one to submit on TIRA platform and perform the test inference. We did it on the early bird submission period and we got **0.755 of accuracy on Spanish and a 0.68 on English**. If we compare it with the results from the experiments, we got more or less an expected result having more accuracy on Spanish than English.

8 Conclusions

Being able to detect fake news spreaders automatically is a really hard task. Although a fully automatic detection of malicious users may still be far, a robust model with a low number of false negatives would have a big impact, providing a first filtering for potential fake news spreaders for humans to later review more carefully, saving their time and resources.

We have seen that the task difficulty also depends on the language, probably not because of the language itself but because of the different ways of making fake news depending on the language that the target people speak. In this case it seems that for the Spanish language it is easier to find patterns to detect fake news.

Despite of the results we have seen that the task is approachable. We are convinced that with a bigger dataset and more research we could get good enough results.

References

[1] Granik, M., Mesyura, V.: Fake news detection using naive bayes classifier. In: 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON). pp. 900–903 (2017)

[2] Pérez-Rosas, V., Kleinberg, B., Lefevre, A., Mihalcea, R.: Automatic detection of fake news. CoRR **abs/1708.07104** (2017), http://arxiv.org/abs/1708.07104

[3] Potthast, M., Gollub, T., Wiegmann, M., Stein, B.: TIRA Integrated Research Architecture. In: Ferro, N., Peters, C. (eds.) Information Retrieval Evaluation in a Changing World. Springer (Sep 2019)

[4] Rangel, F., Giachanou, A., Ghanem, B., Rosso, P.: Overview of the 8th Author Profiling Task at PAN 2020: Profiling Fake News Spreaders on Twitter. In: Cappellato, L., Eickhoff, C., Ferro, N., Névéol, A. (eds.) CLEF 2020 Labs and Workshops, Notebook Papers. CEUR-WS.org (Sep 2020)

[5] Ruchansky, N., Seo, S., Liu, Y.: CSI: A hybrid deep model for fake news. CoRR **abs/1703.06959** (2017), http://arxiv.org/abs/1703.06959

[6] Shu, K., Sliva, A., Wang, S., Tang, J., Liu, H.: Fake news detection on social media: A data mining perspective (2017)

[7] Yang, Y., Cer, D., Ahmad, A., Guo, M., Law, J., Constant, N., Ábrego, G.H., Yuan, S., Tar, C., Sung, Y., Strope, B., Kurzweil, R.: Multilingual universal sentence encoder for semantic retrieval. CoRR **abs/1907.04307** (2019), http://arxiv.org/abs/1907.04307

[8] Yoshua Bengio, Réjean Ducharme, P.V.C.J.: A neural probabilistic language model. Journal of Machine Learning Research **3**, 1137–1155 (2003)