Development of Software for Fake News Detection

Olga N. Kaneva
Omsk State Technical University
Omsk, Russia, Mira Ave. 11, 644050
onkaneva@omgtu.ru

Aleksandr I. Goncharenko
Omsk State Technical University
Omsk, Russia, Mira Ave. 11, 644050
kesha787898@gmail.com

Abstract

During the research process, a fake news detection approach has been developed. The proposed algorithm includes a FastText-based vectorizer, a clustering algorithm, and a set of fully connected neural networks. The algorithm is developed in Python using machine learning libraries and an API is created with the frameworkFlask. The software package is available on the Internet and is accessible to users.

1 Introduction

The relevance of the study is conditioned by the excessive amount of unreliable information on the Internet.This is confirmed by the actions of the authorities of various countries. For example, the government of the Russian Federation passed a package of federal laws on unreliable news in 2019 [Suk2017]. These laws outlaw the publication of false information of public significance spread as credible reports and subject the author of the fake news to an administrative fine, as well as the blocking of the resource where the news was published. Moreover, in the Russian Federation, there are already articles 207.1 and 207.2 of the Criminal Code, which incur criminal liability. The governments of other countries are also interested in this problem. The UK plans to spend more than \$ 22 million to combat fake news in Eastern Europe [Fat2018].

2 Data collection and preparation

For any machine learning task, one of the main steps is to collect and prepare data for model training. Almost all machine learning models have specific features imposed on the input data. The main limitation is the need to represent data in numerical format, preferably as real numbers ($float$, *double*).

To solve the problem, a fairly large dataset was assembled by combining several smaller datasets. These datasets were downloaded from the platform *Kaggle*. A dataset from the hackathon *LocalHackDay* was also used.

As a result, the obtained dataset includes such topics as coronavirus, politics, hostilities, misinformation and Donald Trump's posts (this category is usually separated from politics, because of the large amount of data and the special emotional colouring of the news).

The resulting dataset has 142,062 news in English and 4 columns with features: headline, text, topic, and target feature, which shows whether the news is fake or not.

As a result of using $fasttext$, a vector of dimension $[300, n]$ was obtained for each text, where n is the number of words in the text. Since each text may have a different number of words, it was necessary to transform this vector into a vector of constant length.

Copyright © by the paper's authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

In: Sergei S. Goncharov, Yuri G. Evtushenko (eds.): Proceedings of the Workshop on Applied Mathematics and Fundamental Computer Science 2021 , Omsk, Russia, 24-04-2021, published at http://ceur-ws.org

CEUR-WS.org/Vol-2928/paper13.pdf

For tasks where the emotional component of the text is important, it is best to average vectors for each feature. The heuristic is that when using embedding, the resulting embedding space holds information about the emotional polarity of each word. Therefore, averaging the polarity, one gets the average emotional polarity of the text.

This method was applied to the obtained vectors, and as a result, a vector with the dimension of 300 was obtained for each text.Further, to preserve more information from the original texts, the principal component analysis was used [Ebi018].

For each text, the method of reducing the dimension by words was applied. One principal component was selected for each text. As a result, a vector with the dimension of 300 was obtained.

Further, the vectors obtained as a result of calculating the average and were concatenated into a vector with the dimension of 600through the principal component analysis.

Then, the vectors with the dimension of 600 were employed for clustering and the use of neural networks.

3 Clustering of texts by headlines

The clustering task belongs to unsupervised learning. Clustering can be used to divide the data into several groups, called clusters. Objects in the same cluster can be called related.Clustering can be considered as an optimization problem where one minimizes a metric that evaluates how well a set is partitioned into subsets.

Clustering is applied in this problem because of the large amount of different data. The heuristic is that it will be easier for a neural network to separate data that has been previously partitioned by a qualitatively different algorithm.Lloyd's algorithm has been chosen as a clustering algorithm; it minimizes the sum of the squares of the intra-cluster distances.

The main idea is as follows. At each iteration, the centre of mass is recalculated for each cluster obtained at the previous step. Then, the vectors are split into clusters again according to the new centre which turned out to be closely based on the chosen metric.

The algorithm ends when there is no change in the intra-cluster distance at some iteration. This happens in a finite number of iterations since the number of possible partitions of a finite set is finite, and at each step the total quadratic deviation decreases; therefore, looping is impossible.

The hyper-parameters of the algorithm are K, the number of clusters, and p, the measure of distance.

To solve this problem it was decided to use a complex metric [Dav2018], combining

1) inertia;

2) CalinskiHarabasz metric;

3) Davies-Bouldin metric;

4) the size of the smallest cluster;

5) the range of cluster sizes.

The first three metrics are responsible for making the clusters look alike, and the last two are in charge of cluster size.

All these metrics were aggregated into a single metric.

During the clustering experiment, the dimension of the data was reduced through the principal component analysis. The components giving 90% dispersion were selected.

The optimization problem was solved by the direct search with the upper restriction of the hyper-parameter, i. e. the number of clusters.

Metrics were calculated for each hyper-parameter value and then aggregated into a single one. The learning was conducted with the $MiniBatchKMeans$ algorithm, making it possible to learn not on the entire dataset, but mini-batches.

For this task, the optimal number of clusters was 8. It was with this value of the hyper-parameter that the final model was trained.

Since a small range of cluster sizes was critical for the solution of the problem, it was decided to combine some clusters.All small clusters were combined into a separate cluster, and objects with low confidence about belonging to the cluster were added. Confidence was derived by predicting the probability of belonging to each cluster.

Clusters 3 and 6 and elements with less than 20 % confidence were combined into a new cluster. The visualization of clusters using the dimension reduction algorithm can be seen in Figure 1.

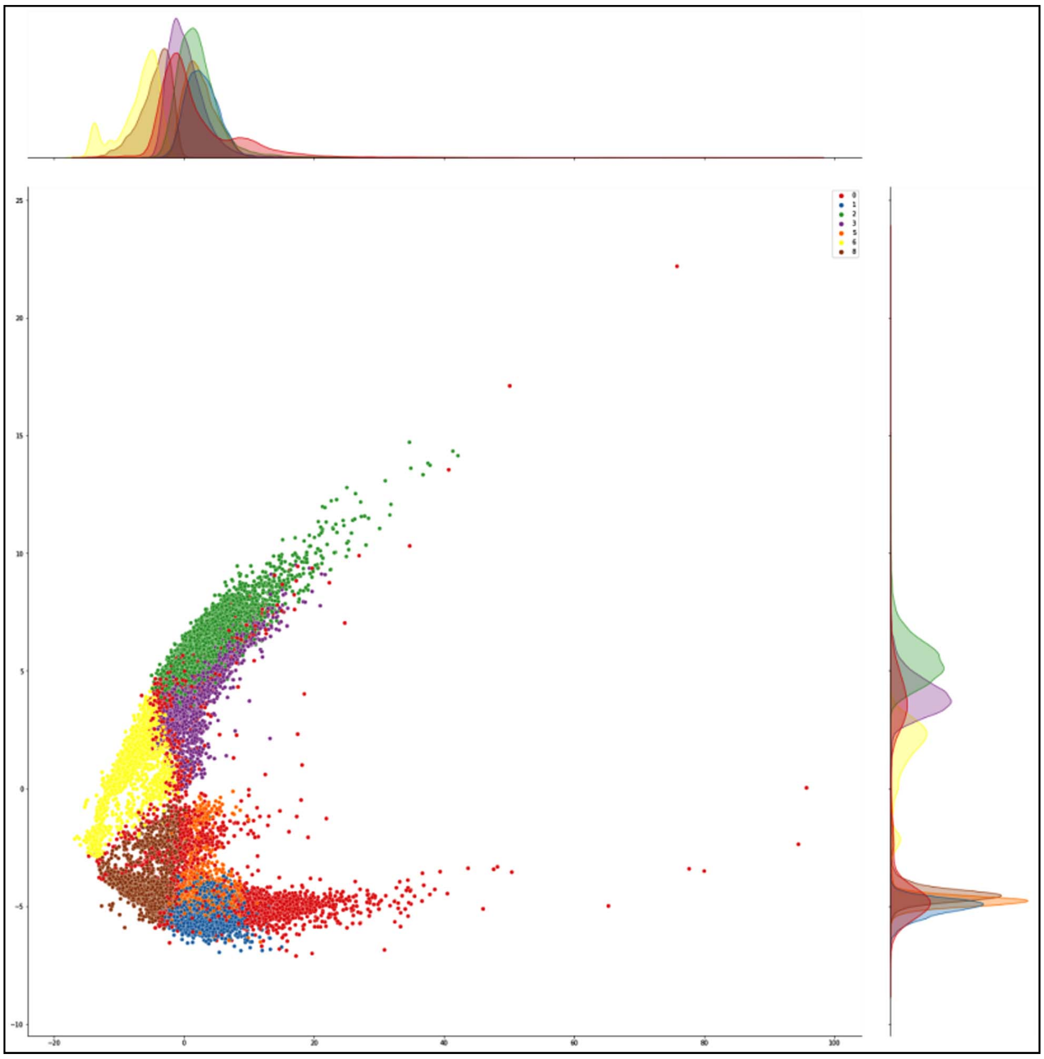


Figure 1: Visualization of clustering

4 Selecting models for each cluster

For each cluster, the neural network must be trained to perform the binary classification task.

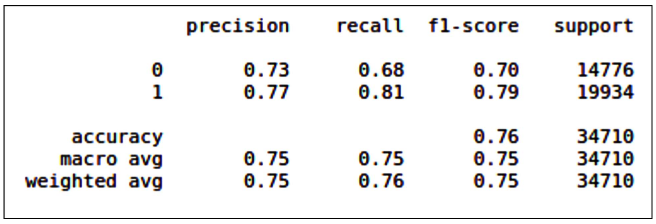
A neural network was chosen as a model because it allows for minimal preparation of features, can work with feature spaces of large dimension, and can extract dependencies of a sufficiently high order. At the same time, fully connected neural networks are quite fast classifiers [Ben2019].

To solve this problem, it was decided to use the architecture of a fully connected neural network consisting of 3 layers with a patch-normalization, a dropout and activation function $PReLU$. Crossentropy was chosen as the $loss$ function, and the $Adamoptimizer$ was used to train the model. Also, weights for classes were used to solve the problem of class imbalance. $F1 - score$ was chosen as the metric.

The results of hold-out validation can be seen in Figure 2. These figures show the precision, recall and $F1 - score$.

5 Embedding in a server application

The resulting models for feature extraction, clustering, and prediction were trained in $jupyter$ notebooks, which is convenient enough to perform experiments. However, this format of writing code does not allow it to be integrated into the application. To solve this problem, the models were saved as files and then loaded into the server application.



```
              precision    recall  f1-score   support

           0       0.73      0.68      0.70     14776
           1       0.77      0.81      0.79     19934

    accuracy                           0.76     34710
   macro avg       0.75      0.75      0.75     34710
weighted avg       0.75      0.76      0.75     34710
```

Figure 2: Metrics of the algorithm

The server application was used to create a simple *API* that allowed any user to utilize the trained models. The *API* also allows other programs to connect to the models even if the third-party programs are not written in *Python*. As a result, this *API* can potentially be used by desktop applications as well as conventional *Websites* and mobile applications.

The framework *Flask* [Gri2014] for the *Python* language was used to create the *API*. It is often referred to as a class of micro-frameworks since it is rarely used to write large applications and is rather intended for creating small frameworks for web applications.

The algorithm reproduced in the *jupyter* notebooks was integrated into the application.

The algorithm itself expects an input of the text and the headline of the news in English. To do this, the resulting entities are cleaned of garbage characters using regular expressions. Next, the lines are checked for non-emptiness.

If both lines are empty, the algorithm sends an empty response, which should be processed on the user's side.

Next, the input data are converted using the written *TextToVec* class, which implements a text conversion algorithm. Firstly, the class divides the text into tokens, each token is converted into a vector using *FastText*, and then the average of the vectors and their first principal component are selected.

Next, if there is a headline in the input data, a cluster is defined, using *Clusterizer* inherited from *Predictor*. This class implements the definition of a cluster depending on the distance to the cluster centres. Additionally, if there is no confidence in belonging to the right cluster, then the object is sent to the garbage cluster and further clustering is not used. Also, since some clusters are combined into one, the dictionary determines the true cluster.

The presence of clustering allows us to add new model data. To do this, it is only needed to find new data, determine their feature, train a neural network on them, and build this cluster into an existing clustering. In this case, it is not necessary to retrain the entire model, which is a time-consuming process.

Next, if the cluster of the data is not defined or it is defined as a garbage one, the neural network trained on the entire dataset is activated.

If there is no problem with the cluster number, then, depending on the number, a model trained on similar data is activated. *TorchPredictor* class is used for this purpose. Probability is received as a result.

This algorithm was successfully integrated into the Flask application.

6 Conclusion

During the research process, a fake news detection approach has been developed. To do this, data have been collected from various sources and then used as a basis for a rather large dataset containing fake news on various topics. A clustering algorithm has been developed, which is based on the algorithm *Kmeans*, which groups vectors of texts according to the closeness of their value. Also, the sizes of the resulting clusters have been improved. A detection algorithm has been developed for each cluster, the algorithm being a set of neural networks trained on the source data. The developed algorithm for detecting fake news has been built into the server application on *flask* and an open *API* has been created.

References

[Suk2017] A. P. Sukhodolov. The phenomenon of "fake news" in modern media space. - Eurasian Partnership: Humanitarian Aspects. *Proceedings of the International Scientific and Practical Conference. Irkutsk: Publishing house of Baikal State University*, 1:93–112, 2017.

[Fat2018] I. A. Fateeva. Social networks in the aspect of media safety. *Media-education 2014: Proceedings of the All-Russian scientific and practical conference (with international participation) "Media-education 2014. Regional aspect" 7 October - 9 October 2014 / Edited by I.V. Zhilavskaya, E.A. Karyagina*, 304-310, 2014.

[Ebi018] H. M. Ebied, M. Yu. Savelev, T. Yu.Fink. Feature extraction using PCA and Kernel-PCA for face recognition. *2012 8th International Conference on Informatics and Systems (INFOS)*, pp. MM-72-MM-77, 2012.

[Dav2018] D. L. Davies, D. W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2): 224–227, april 1979.

[Ben2019] B. Bengfort, R. Bilbro, T. Ojeda. Applied Text Analysis with Python. *AIP Spb: Peter*, 2019.

[Gri2014] Flask Web Development / Edited by Miguel Grinberg. O'Reilly/DMK Press, 2014.