In Search of the Real Fake News

Jaakko Kuurne, Teemu Pöyhönen, Dean Rahman

Introduction to Data Science: Final Assignment

first.last@helsinki.fi

2020 Autumn

1 Introduction

We have built a text classification system for predicting whether a news article is real or fake. We have presented our motivation for choosing to work on this kind of system and our longer term vision in our promotional pitch (Kuurne et al., 2020a). We have also written a blog (Kuurne et al., 2020b) for our target user who is primarily someone who investigates whether news is fake. But it could also be anyone who reads the news and wants to be able to quickly paste in a story and get an indication whether the story is fake. Without going into details there on the algorithms we used, we have provided that audience an overview of how we built our system and explained how we assessed its performance. This enabled us also to present our performance results there. We believe that truth investigators would want to understand our work at this level of detail in order to have the confidence to use our system's automatically generated fakeness assessments.

The goal of this technical report, on the other hand, is very much to detail how we have built a Model-A and a Model-B. They are both logistic regression models where Model-A uses a TF-IDF feature set. For Model-B, we first considered a word2vec feature set, but then learned about and quickly moved on to using a Doc2vec feature set instead.

After briefly discussing the justification for using logistic regression, we present our data pre-processing methods. Then, we discuss the two different input feature sets for the model. Finally, we discuss some future work.

2 Logistic Regression

The nature and form of the outputs of different models influenced our choice of model architecture. We chose logistic regression because it produces a probabilistic output and this is quite interpretable by an end user. We avoided, for example, random-forest classifiers because of the lack of end-user interpretable output even though such classifiers might have provided better performance.

Another reason for using the logistic regression model is the rather accessible library of scikit-learn. We were able to focus on tuning the hyper-parameters and processing the data because importing the model and fitting the vectors to the model could be done without much effort. Our logistic regression model uses the liblinear solver, balanced class weight and the

one-versus-rest scheme (ovr).

3 Data

Our data came with only some minor pre-processing (lower-casing and punctuation removal). We spent considerable effort on more sophisticated data pre-processing methods. We experimented with stemming and lemmatization, and for different reasons neither was used in either model. In the case of Doc2vec, morphosyntactical information regarding words and sentences is an important aspect when generating vector representations of documents. Within our TF-IDF model, lemmatization could been a useful pre-processing method, but our hardware was the limiting factor.

Before training the Doc2vec representations, we did convert all URLs to a 'URL' token. We tokenized the documents using nltk's RegexpTokenizer.

4 Input Features

4.1 Doc2vec

Representation learning and word2vec are most likely already familiar to any practitioner of machine learning and natural language processing methods. We employed Doc2vec as a method to input vector representations of the news into the logistic regression model. Doc2vec differs from word2vec in that Doc2vec adds the paragraph id as a vector to the typical word2vec model. Hence, we got a word vector for each of the words and a document vector for each of the documents. We then trained the model by "predicting a probability distribution of words in a paragraph given a randomly-sampled word from the paragraph."(Susan Li)

4.1.1 Doc2vec Implementation

We divided the news articles into a test set and a train set with the test set containing 20% of the articles. Then, we trained the Doc2vec model, and as each of the news articles were tagged with a dummy index, we were able to retrieve the training and test vectors after the training process. Finally, we could retrieve the vectors and used them as input features for the logistic regression model.

We used a Distributed-Bag-of-Words version of the Paragraph Method (PV-DBOW) which is comparable to the skip-gram method in word2vec.

t

4.1.2 Doc2vec Parameters

In our methodology, we used the Gensim library for Doc2vec. The trained Doc2vec model used vector-size 300. The size of the dimensions can be seen as the capacity of the embedding model to capture information. Nevertheless, increasing the dimensionality above 300 was not appropriate in this project, since the size of the dataset was not large enough to warrant a higher dimensionality.

We set the window size to 15 to try to capture the semantic similarity of words and not strictly the syntactic similarity. The model still leaned towards syntactic information as the window size for a purely semantic approach could go as high as 50.

We set negative sampling to 5 though we experimented with larger numbers, such as 15. Fifteen seemed a more natural choice based on the size of the dataset and it was relatively fast to train the embeddings even with the higher negative sampling. However, the results did not show improvement.

Minimum count was another important parameter, since our dataset contains many infrequent and irrelevant words, e.g., the text of navigation buttons in some of the articles. We did attempt to clean the articles from this kind of noise, but the simplest solution under our time constraint was to drop words such as usernames with minimum count. We experimented with values below 5, but even with 5 the model did not seem to lose any important words.

For the final model, we set the epoch count to 50 and it took almost 5 hours to train the embeddings.

Lastly, we set a learning (and minimum) rate to 0.01. We did investigate a learning rate as high as 0.5, but 0.01 seemed to yield similar if not better results.

4.2 TF-IDF

Term Frequency - Inverse Document Frequency (TF-IDF) was our other approach for constructing features. TF-IDF turns textual documents such as our news articles into numeric representations. Then, a machine learning

model such as logistic regression is able to take the numeric representations as an input.

Each instance (news article) is a collection of features. In the TF-IDF scheme, there exists an equal amount of features as there are unique words in the vocabulary. The value of each feature is the TF-IDF score for that particular word.

$$\text{tf-idf}(t, d) = \text{tf}(t, d) * \text{idf}(t), \text{ where idf is defined as follows:}$$

$$\text{idf}(t) = \log(n/\text{df}(t)) + 1$$

The underlying intuition behind TF-IDF is that it gives more weight to words that are common in a particular document and less weight to frequent words that occur in other documents. Consequently, TF-IDF produces a higher score for words that are frequent in a particular document and do not occur often in others.

4.2.1 TF-IDF Implementation

We used the TF-IDF implementation of the python package scikit-learn(sci). We decided to test the results using TF-IDF as a way of constructing our features because we thought that the importance as defined in the context of TF-IDF might help the model to differentiate between fake news and real news. Additionally, TF-IDF is accessible to understand as a concept and is relatively straight-forward to implement. For these reasons, TF-IDF is a decent candidate as a baseline when we compare the performance with other feature extracting techniques.

The disadvantage of TF-IDF is that it creates feature vectors from a large corpus that are highly dimensional. The feature vector is a V dimensional vector where V equals the size of the vocabulary. Fortunately, scikit-learn can handle sparse matrices efficiently.

For the purposes of further examining the data and doing sanity checks, we also implemented a function that checks what documents contain a target feature. In other words, the function finds documents in which a particular feature has a non-zero TF-IDF score.

4.2.2 TF-IDF Parameters

We used a minimum document frequency of two to develop the feature vectors. So we ignored any word that appears only in a single document. We also set the n-gram range to 1 so only individual words were considered independently. We did experiment with an n-gram range of '1, 2' in which both single words and all pairs of adjacent words were considered, but this did not seem to improve the performance of our model. Furthermore, it increased the number of features greatly, so we kept an n-gram range of 1.

5 Future Work

In order to improve the performance of the logistic regression model itself, we would research different methods for pre-processing data. In the original dataset, some articles were actually a series of blog posts, with comments and discussion and even text of user interface elements. We already implemented a script that can fetch a large number of html documents relatively quickly via asynchronous processing of requests. By creating a heuristic or employing an existing one to pick the relevant text body from article pages, we might be able provide a cleaner dataset to build our features.

As another example, lemmatized tokens could be an important improvement in terms of pre-processing and building the logistic regression model with TF-IDF features. This would allow different variants of the same word (e.g. politics, political, politicize) to contribute to combined TF-IDF scores to more strongly distinguish an article (e.g. about politics) from other articles in the dataset (not or less about politics). Similarly, this could strengthen the distinction of words or topics more commonly in fake news.

In terms of seeking improvement in the word and document embeddings as features, we would be exploring openly available word embeddings that have been pre-trained using massive amounts of data (e.g. Wikipedia). We would continue to consider training our own embeddings as well, using the data we have collected. This is in case using our own embeddings is more conducive to growing our training set as we collect data from more and more truth organizations.

Lastly, we want to explore deep learning, especially to replace the logistic regression classifier with, for example, a zero-shot Transformer model.

References

TfidfTransformer. URL https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html.

J. Kuurne, T. Pöyhönen, and D. Rahman. In search of the real fake news pitch, 2020a. URL https://jjaakko.github.io/fake-news-classifier/assets/In_Search_of_the_Real_Fake_News.pdf.

J. Kuurne, T. Pöyhönen, and D. Rahman. In search of the real fake news blog, 2020b. URL https://jjaakko.github.io/fake-news-classifier/.

Susan Li. Multi-class text classification with doc2vec logistic regression. URL https://towardsdatascience.com/multi-class-text-classification-with-doc2vec-logistic-regression-9da9947b43f4.