Research report - Hyperpartisan NLP

Florian Biebel Daan Middendorp

May 2019

1 Introduction

This report is part of a project at the Technische Universität Berlin, called Advanced Projects at the Quality and Usability Lab. During this project, the problem of hyperpartisian news detection will be solved using natural language processing. Hyperpartisian news is news that is extremely biased or sharply polarized in favor of a political party.

This problem comes from a task in the Semantic Evaluation[1] competition which is held simultaneously during this project. This makes it possible to use the dataset provided to the participants of the competition. The goal of this project is to achieve a deeper understanding of the inner working of such a system and learn how to build such a system ourselves.

2 Datasets

The competition provides different datasets with different sizes and different quality. The first dataset, which is called *byarticle* is manually classified and contains about 600 articles.

The second dataset, which is called *bypublisher*, contains over 600.000 articles and is classified based on the publisher. This makes the the quality of the dataset worse, but for systems where a lot of training data is needed, it might be an advantage.

The problem with such large dataesets, is that there is a lot of computational power needed in order to train a network using this data. Therefore, during this research, a third dataset, called *byart-bypub-mix* was created. This dataset consist out of the first *byarticle* dataset with another 1100 articles from the second dataset added.

[1] `https://pan.webis.de/semeval19/semeval19-web/`

3 Existing approaches

3.1 Preprocessing

The dataset from SemEval contains a thousands of articles. These articles cannot be fed into a form of artificial intelligence immediately. Several approaches are using the following preprocessing:

3.1.1 Tokenization

Tokenization is the process of dividing text into sentences and sentences into words and punctuation marks. From tokenized text, further preprocessing can be applied.

3.1.2 Stemming

Stemming is the process of removing inflection from words. Usually only the stem is necessary information to abstractly understand context. An example is reducing "stemming" and "stems" to "stem".

3.1.3 Lemmazation

An alternative method to stemming. In exchange for speed, the accuracy of removing inflection improves. This greatly improves accuracy for texts with multiple instances of words with similar inflections. An example is "leaf/leaves" and "leave/leaves".

3.1.4 Stopwords

Stopwords are words that do not influence the context or meaning of a sentence. For example, words such as "the", "and", and "or" can usually be considered stopwords. Filtering out stopwords improves accuracy and performance due to removing unnecessary words.

3.2 LSTM

Long short-term memory a form of an artificial neural network. The difference with a recurrent neural network is that LSTM networks are able to use information from the past to understand the input. E.g. a video frame can be better understood if the previous video frames are also taken into account. This is also valueable for text processing, because hyperpartisian news detection requires a deeper understanding of the link between words.

3.3 CNN

Convolutional neural networks have proven to be useful for multiple NLP tasks, such as sentiment analysis or summarization. CNN have the ability to extract

n-gram features from input sentences to create a semantic sentence representation to be used in downstream tasks. Such ability comes from the convolutions creating an evermore abstract representation of the input, but still conserving a micro-context due to how convolutions work. This particular usage of CNN is called Sentence Modeling and could prove useful for hyperpartisan news detection.

3.4 Naive Bayes

The Naive Bayes method utilizes Bayes' Theorem to classify text and is used in text and topic classification. Additionally, it relies on the 'bag of words' (BOW) representation to function. For the binomial classification a positive and negative class are used—in our case hyperpartisan and non-hyperpartisan—to each of which is a BOW assigned. Depending on the count of positive and negative words, the text is classified as such. This approach might be useful depending on the chosen BOWs.

3.5 Support Vector Machine

If the wordt are modelled as vectors, is is possible to draw them in a multidimensional space. Support Vector Machines are able to draw a line in the space with creates the best separation between two types of categories. Therefore it is really suitable as a binary classifier. In this case both classes would be *hyperpartisian* or *non-hyperpartisian*.

3.6 Logistic Regression

Logistic regressions are similiar to Support Vector Machines. However, line that is drawn in order to separate two classes is calculated in a different way. A support Vector Machine tries to draw this so that it maximizes the margin between the line and both classes. Logistic Regression is therefore more sensible to outliers. Another difference is that Support Vector Machines are binary, so there is no way to tell something about the reliability of the prediction, the class is either 1 or 0. Logistic Regressions are working with probabilities between 1 and 0

4 Classifier

4.1 Sckitlearn

Skikitlearn is a software library which can be used for machine learning. It can be used with Python and provides a lot of tools that can be used out of the box.

4.1.1 Preprocessing

The data provided by the SemEval contest is stored as XML. Therefore the data had to be parsed. Inside the articles, there was also a lot of normal HTML tags included. The tokenizer would also include these tags, so these had to be stripped.

After the data is converted to plain text, it is sent to the transformer. For this research two different types of transformers are used: *CountVectorizer* and *TfidfTranformer*. The first one uses the absolute value for the number of times that a term occurs. The second one also takes the general frequency of the term in to account. This means that terms that are really common, like stopwords, are considered less important. The output of this process results in a *bag of words*-model.

4.2 Classifier

These bags of words can be feeded to the actual classifier. In this research, the following classifiers are experimented with.

- Logistic regression
- SGD Classifier
- Random Forest
- Naive Bayes

After some tweaking of the paramters, these classifiers resulted in quite good results.

Method	Acc.	Prec.	Rec.	F1
Logistic regression	0,70-0,82	0,65-0,81	0,38-0,60	0,51-0,70
SGD Classifier	0,62-0,76	0,51-0,79	0,52-0,78	0,59-0,70
Random Forest	0,63-0,78	0,60-0,88	0,27-0,47	0,41-0,59
Naive Bayes	0,65-0,78	0,50-0,75	0,57-0,80	0,59-0,71

However, it seemed hard to make these scores much better by further tuning. Because of this, the next step of the research covered the tuning of the preprocessing.

During manual investigation of the dataset, it turned out that a couple of metrics of the articles are different based on the category. For example, an article is much longer if it is an hyperpartisian article. The difference is 596 words vs. 415 words.

Also the use of capitalized words is extravagant. Hyperpartisian articles have an average of 7 capitalized words, vs. 5 capitalized words per article for normal articles.

At last, the manual research also turned out that the use of exclamation marks is more dense in hyperpartisian articles. The average lies at 1 exclamation mark per article vs. 0.36 for normal articles.

In order to use this metrics for the classifier, these metrics had to be embedded in the preprocessing. The idea was to add one token, e.g. LONGARTICLE, if an article had over 500 words. Unfortunately, this did not result in significant better scores.

The second attempt used a so called *feature union*. This makes it possible to combine multiple pipes into one classifier. The disadvantage, is that this happens after the preprocessing, so there is no way to recall the original article an calculate the metrics. The only thing that can be determined based on the preprocessed data is the length of an article (because of size of the bag of words). This feature union was created, and resulted in a slightly improved score:

4.3 Performance

Method	Acc.	Prec.	Rec.	F1
Logistic regression	0,70-0,82	0,65-0,81	0,38-0,60	0,51-0,70
SGD Classifier	0,62-0,76	0,51-0,79	0,52-0,78	0,59-0,70
Random Forest	0,63-0,78	0,60-0,88	0,27-0,47	0,41-0,59
Naive Bayes	0,65-0,78	0,50-0,75	0,57-0,80	0,59-0,71
Log. reg. FU	0,71-0,82	0,67-0,92	0,39-0,64	0,49-0,70
SGD FU	0,34-0,70	0,00-0,40	0,00-1,00	0,0-0,58

5 conclusion