**Classical Machine Learning Algorithms for Classifying Protest Related News**

**Fatih Beyhan**

[beyhanf@outlook.com](mailto:beyhanf@outlook.com)

**Abstract**

In this paper, different types of algorithms are being used for classifying p-rotest related news in a small dataset and their results are shown. Some challe-nges about dataset and task are being handled. Support Vector Machine has b-etter results than advanced approaches on this specific task due to properties of the dataset. Advanced algorithms are discussed. This report will not only explain which algorithms and approaches did work on this task but also which algorithms did not work.

**Keywords:** protest classifier, news, machine learning, deep learning.

1. **Introduction**

Text classification is one of the main tasks in NLP field. There are many different approaches and methodologies for given dataset and classification type. One of the most challenging parts in these tasks is to find the right dataset for desired classification type, such as classifying news whether they are related to a protest or not.

In our task a small dataset, which consist of URLs from The Hindu and labels, is given. It is obvious that this is a binary classification task. This project has two main parts; data extraction and “learning” to classify. Code can be found in the author’s GitHub page[[1]](#footnote-1).

1. **Dataset**

As it is mentioned above, dataset consist of URLs and their labels. There are 774 URLs in this list. To begin the main purpose of this project, which is classifying the news, extraction of the news is needed. Even the URL of the news might tell us some about it. The URLs are mostly consist of 3 main parts; section, location and headline. There are also article ID which can be useful for achieving purposes. The location and time components of the news can be correlated for the protest news since the protests may take long time. However, a valid way to integrate this information to the models is not found, yet. For the extraction, at first [BeautifulSoup](https://www.crummy.com/software/BeautifulSoup/bs4/doc/) library of Python was used. Even it is working fine, extraction of the related data from the HTML document was tedious. Then another library, [newspaper](https://newspaper.readthedocs.io/en/latest/), was selected for this purpose. This library made the extraction of the content much easier. Some of the articles could not be extracted properly with this library, however, the ratio is quite small.

Beside the content, the section and location of the news and their headlines were extracted and the raw data was saved as another csv file for further process. There is another challenge on this dataset. The dataset is imbalanced. Only the 195 of the 774 news are protest news. Different approaches were tried to handle this problem which will be mentioned in the further chapters.

The final and supreme challenge of this dataset is that it’s coming from only one source and targeted news are coming from different sources. Although, classic machine learning problems assume that the dataset is a proper representation of the population, this assumption is not fulfilled.

1. **Preprocessing and Analysis**

After the extraction, our data has 5 parts; section, location, headline, content and label. Headline and content were merged into single column. Section and location parts were not used for this project. After this process, there are content and label columns.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Avg. # of Sent.** | **Avg. # of Words** | **Max & Min # of Sent.** | **Max & Min # of Words** |
| **Protest News** | 15.13 | 225.50 | 60 & 3 | 1520 & 29 |
| **Non-Protest N.** | 19.82 | 263.93 | 170 & 3 | 1245 & 22 |
| **All News** | 18.63 | 254.25 | 170 & 3 | 1520 & 22 |

Table 1: Average, max and minimum number of the words and sentences per article for protest news, non-protest news and all news.

Average, max and min number of the words and sentences per article for protest news, non-protest news and all news before the preprocessing can be seen on **Table 1**. Basically, whether an article is about a protest or not will be predicted from approximately 19 sentences and 254 words.

The most common 10 words and their ratio for protest news and non-protest news are can be seen on Table 2.

|  |  |  |
| --- | --- | --- |
| **Protest News** | **Non-Protest News** | **All News** |
| ‘state’ - 0.43 | ‘government’ - 0.25 | ‘government’ - 0.30 |
| ‘government’ - 0.43 | ‘state’ - 0.24 | ‘state’ - 0.28 |
| ‘district’ - 0.42 | ‘year’ - 0.24 | ‘district’ - 0.24 |
| ‘protest’ - 0.41 | ‘take’ - 0.21 | ‘year’ - 0.22 |
| ‘police’ - 0.40 | ‘time’ - 0.21 | ‘minister’ - 0.21 |

Table 2: The most common 5 words for each class and for the dataset can be seen with their ratios among the given class. Ratio indicates the number of the unique article that has the word divided by whole population of the given class.

Even before the machine learning models we can do some prediction based on the vocabulary of the article with Bayes’ Theorem.



Equation 1: Probability of class K for given X.

With this formula we can statistically predict whether an article is about a protest or not, i.e. let’s say the word “protest” exists in an article. After the numbers are inserted in this equation, Bayes’ Theorem says if an article has “protest” word in it, it is 80% related to the a protest, surprisingly not 100%. The “word affect” of other words can be calculated as well. However, we are not interested in statistical approach on this task, since the dataset is really small and most probably not able to represent the population, hence, our statistical model would not be generalizable.

Classical machine learning algorithms cannot work with strings. These news have to be represented by some numbers while keeping them differentiable. The main purpose of this is to represent each article in the same multidimensional space with vectors. TfIdf vectorizer of [scikit-learn](https://scikit-learn.org/stable/) library of Python was used for this purpose. Right before TfIdf dataset has to be cleaned to get more efficient representations. Hence, punctuations, numbers and stopwords were thrown away and whole news were lowercased. Then remaining words were lemmatized by WordNetLemmatizer of [NLTK](https://www.nltk.org/) library. These processes decreased the number of the words.

Models were Pipeline objects with two components; TfIdf and classifier. 3 different classifier algorithm, which are LogisticRegression(), SVC() and KNeighborsClassifier(), was used. The results of KNN will not be shown since it did way worse than other two. The main purpose of selecting these three algorithms is that, each of them use different way of classify and each algorithm could catch some mistakes of other.

Finally, the dataset were split into train and test parts with 0.3 ratio.

1. **Models and Results**

After making our dataset bunch of numbers, algorithms are ready to take these numbers and give results. This section will be in two parts; classical machine learning algorithms and “what did not work”.

* 1. **What did work?**

For classical approaches, GridSearchCV method of [scikit-learn](https://scikit-learn.org/stable/) library was used for hyper-parameter tuning. F1-Macro was selected as GridSearchCV scoring option. So, GridSearchCV will bring the best parameters for the model that gives the maximum F1-Macro score. The reason we are focusing on F1-Macro is that we have an imbalanced data. Even if the classifier would classify each instances as class 0 then our accuracy would be almost 75%. However F1-score makes sure that each class is being correctly classified. F1-Macro is basically the average of F1-score for each class. We do not want anything weighted because we have an imbalanced dataset.

The best parameters for LogisticRegression() pipeline were;

**{'clf\_\_C': 15, 'tfidf\_\_min\_df': 4, 'tfidf\_\_ngram\_range': (1, 2)}**

Due to unbalance of the dataset, it gave different weights to the classes. With these parameters the classification report of the model on test set is given in the **Table 3.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| **0.0** | 0.88 | 0.98 | 0.92 | 174 |
| **1.0** | 0.90 | 0.59 | 0.71 | 59 |
| **Accuracy** |  |  | 0.88 | 233 |
| **Macro avg.** | 0.89 | 0.79 | **0.82** | 233 |
| **Weighted avg.** | 0.88 | 0.88 | 0.87 | 233 |

Table 3: Classification report of logistic regression on test set.

The best parameters for SVC() (support vector classifier) pipeline were;

**{'clf\_\_C': 5, 'clf\_\_gamma': 'scale', 'clf\_\_kernel': 'linear',**

**'tfidf\_\_min\_df': 4, 'tfidf\_\_ngram\_range': (1, 2)}**

As it was in logistic regression, class weights are different due to unbalance of the dataset and the kernel of the model is linear. With these parameters the classification report of the model on test set is given in the **Table 4.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| **0.0** | 0.90 | 0.98 | 0.94 | 174 |
| **1.0** | 0.91 | 0.69 | 0.79 | 59 |
| **Accuracy** |  |  | 0.91 | 233 |
| **Macro avg.** | 0.91 | 0.84 | **0.86** | 233 |
| **Weighted avg.** | 0.91 | 0.91 | 0.90 | 233 |

Table 4: Classification report of SVC on test set.

The results of the SVC is better than logistic regression.

To improve the reducible error due to the imbalanced data, data was under-sampled. New data had 195 instances for each class and balanced class distribution obtained. After this process GridSearchCV was used for both logistic regression and support vector classifier.

Best parameters for logistic regression was:

**{'clf\_\_C': 15, 'tfidf\_\_min\_df': 4, 'tfidf\_\_ngram\_range': (1, 3)}**

**}**

With these parameters the classification report of the model on test set is given in the **Table 5.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| **0.0** | 0.89 | 0.86 | 0.87 | 63 |
| **1.0** | 0.84 | 0.87 | 0.85 | 54 |
| **Accuracy** |  |  | 0.86 | 117 |
| **Macro avg.** | 0.86 | 0.86 | **0.86** | 117 |
| **Weighted avg.** | 0.86 | 0.86 | 0.86 | 117 |

Table 5: Classification report of logistic regression on under-sampled test set.

Compared to the original data, scores are improved. This results shows that under-sampling helped logistic model to make better predictions.

The same procedure was implemented for SVC. The best parameters for SVC was:

**{'clf\_\_C': 1.0, 'clf\_\_kernel': 'sigmoid',**

**'tfidf\_\_min\_df': 4, 'tfidf\_\_ngram\_range': (1, 2)}**

The results of the SVC model with these parameters on under-sampled test data is shown in **Table 6**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| **0.0** | 0.89 | 0.86 | 0.87 | 63 |
| **1.0** | 0.84 | 0.87 | 0.85 | 54 |
| **Accuracy** |  |  | 0.86 | 117 |
| **Macro avg.** | 0.86 | 0.86 | **0.86** | 117 |
| **Weighted avg.** | 0.86 | 0.86 | 0.86 | 117 |

Table 6: Classification report of SVC on under-sampled test set.

After all, SVC() is doing slightly better than LogisticRegression and models that are trained on original data are giving better results compared to the models that trained on imbalanced data. For the classical approaches, SVC() is the selected model. The final test set results of SVC() and ensemble model (LogisticRegression, SVC and KNN) is shown on **Table 7** and **Table 8**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| **0.0** | 0.93 | 0.95 | 0.94 | 943 |
| **1.0** | 0.51 | 0.41 | 0.45 | 107 |
| **Accuracy** |  |  | 0.90 | 1050 |
| **Macro avg.** | 0.72 | 0.68 | **0.70** | 1050 |
| **Weighted avg.** | 0.89 | 0.90 | 0.89 | 1050 |

Table 7: Classification report of SVC() on last test set.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1-Score** | **Support** |
| **0.0** | 0.93 | 0.97 | 0.95 | 943 |
| **1.0** | 0.58 | 0.34 | 0.43 | 107 |
| **Accuracy** |  |  | 0.91 | 1050 |
| **Macro avg.** | 0.75 | 0.65 | **0.69** | 1050 |
| **Weighted avg.** | 0.89 | 0.91 | 0.90 | 1050 |

Table 8: Classification report of ensemble model on last test set.

* 1. **What did not work?**

After implementing classical approaches, advanced models were tried and, unfortunately, failed.

The first approach was fully connected neural networks. A single article was represented by 1870 dimensional vector. That means the input of the NN will be 1870. Even if a perceptron was used, there will be 1870 weights to estimate with only 606 instances. Inevitably, NN will overfit. This is basically, trying to find 1870 unknowns with only 774 equations. A simple perceptron with sigmoid activation gave the result on the **Table 9.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-Score | Support |
| 0.0 | 0.74 | 0.93 | 0.82 | 27 |
| 1.0 | 0.92 | 0.72 | 0.81 | 32 |
| Accuracy |  |  | 0.81 | 59 |
| Macro avg. | 0.83 | 0.82 | **0.81** | 59 |
| Weighted avg. | 0.84 | 0.81 | 0.81 | 59 |

Table 9: Classification report of Perceptron on under-sampled test set.

Results shows that NN is not better than classical algorithms and overfits. The learning curve says it all.

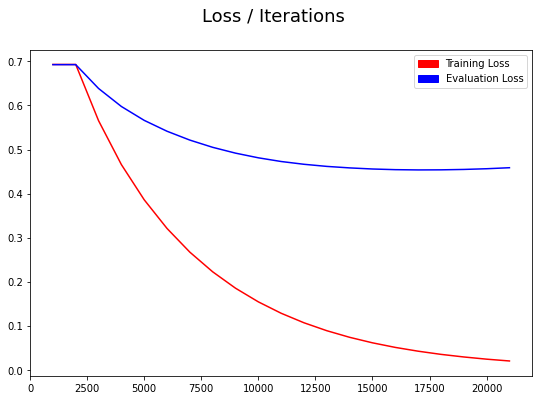


Figure 1: Learning curves of the perceptron model. It is obvious that this model overfits.

Neural Networks with hidden layers were used as well but gave worse results, so, it is not going to be shown here.

To solve the problem with high dimensionality, PCA was used and with 230 components, 90% of the variance in the 1870 dimensions were captured. However, it did not work inconceivably. Scores were worse than previous approaches.

Another approached was to classify news by LDA, which is an unsupervised method. Despite the fact that LDA was not doing a good job on the classifying task, the probabilities its assigning to each class for each article could’ve been useful. So these scores were added on the dataframes for each approach. But again, this did not work and even made predictions worse.

Doc2Vec algorithm was implemented to try different representation of articles and a NN was used to classify news but this method also did not give good results on the dataset.

The most advanced algorithm of this project, and current NLP world, was applied: BERT. Since the dataset is extremely small for training a transformer from scratch, a pre-trained model, BertForSequenceClassification, was used. The implementation was inspired by a blog from <http://ohmeow.github.io/>. It uses huggingface transformers and fastai tools and integrates these two with blurr library. In short, this approach did not do well as well.

The dumbest algorithm of this project was selecting a sample of words and using only these words to classify news. Words were selected by Bayes’ Theorem. Each word has its own effect on classifiers. As the example given above, if an article had “protest” word in it, then this article is 80% protest related. There were 41 words that are assigning higher than 60% protest-related probability. Each document were represented by 41 dimensional binary vector and this data was fed into SVC() and gave almost same results with SVC() of whole dataset. Nevertheless, this approach might have some generalizability issues.

1. **Conclusion**

The main challenge on this problem is the feature space is not definite. With the each new word, the feature space is getting expanded. The perfect solution would be finding efficient and useful dataset that will provide the best feature space which will make the model generalizable. Another interesting thing that with only 41 words, which is much less than what our models are using, it is possible to get scores close to the best scores.

With a small and imbalanced dataset, the best approach was the logistic regression with original dataset for protest classifier. Although transformers are sweeping the all tasks on NLP, it did not help on this specific dataset, however, a larger and healthier dataset would make BERT model give excellent results. Nevertheless, if our dataset is divisible by simple models, there is no need to use complex models. Nobody wants to cut a bread with a lightsaber.

1. **Comments and Further Work**

After reading papers from CLEF 2019 on this task and seeing our results on test dataset from different source, it’s clear that the main challenge of this task is the generalizability of the models. There are several different source and dozens of contexts. To set an example for better understanding of the problem, what we are trying to do is, trying to understand the teenager behavior from only sample of teenagers from one country. It is possible to get some ideas but coming up with general understanding of teenager behavior, we need wider sample and deeper models.

We could’ve used different classifier, such as XGBoost, AdaBoost, etc. but the line between underfitting and overfitting is really thin on this project and we prefer to underfit rather than overfit. The idea behind simple models is that they will get almost same results on any kind of source. For this purpose, even the given parameters to GridSearchCV were restricted.

Human-beings are able to detect the protest news clearly, even if they are coming from different sources. That means deeper understanding of articles would help to improve the models. Hence, WordEmbeddings would be the next step of this project. They will help the models to consider the semantic relationships between the articles. With WordEmbeddings and maybe bigger dataset, deep learning models would give better results.

**REFERENCES**

* Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. (2013). *An introduction to statistical learning: with applications in R*. New York :Springer
* Hürriyetoğlu, A., et al. *Overview of CLEF 2019 lab ProtestNews: extracting protests from news in a cross-context setting*. In: Crestani et al. [94]
* Safaya, A. *Event Sentence Detection Task Using Attention Model.* ProtestNews Lab at CLEF 2019.

1. <https://github.com/fatihbeyhan/ProtestClassifier> [↑](#footnote-ref-1)