Classical Machine Learning Algorithms for Classifying Protest Related News

Fatih Beyhan

beyhanf@outlook.com

Abstract

In this mini-task, different types of algorithms are being used for classifying protest related news in a small dataset and their results are shown. Some challenges about dataset and task are being handled. Classical approaches have better results than advanced approaches on this specific task due to properties of the dataset. 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].

2. Dataset

As it is mentioned above, dataset consist of URLs and their labels. There are 606 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 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, was selected for this purpose. This library made the extraction of the content much easier. 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 unbalanced. Only the 146 of the 606 news are protest news. Different approaches were tried to handle this problem which will be mentioned in the further chapters.

[1] https://github.com/fatihbeyhan/ProtestClassifier

3. 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	15.23	229.24	60 & 3	1520 & 29
Non-Protest N.	20.28	270.92	170 & 3	1245 & 22
All News	19.01	260.45	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 260 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
‘district’ - 0.4383	‘government’ - 0.2736	‘government’ - 0.313
‘government’ - 0.4315	‘year’ - 0.2483	‘state’ - 0.285
‘state’ - 0.4246	‘state’ - 0.2391	‘district’ - 0.244
‘protest’ - 0.3972	‘take’ - 0.2253	‘year’ - 0.2271
‘police’ - 0.3698	‘time’ - 0.2138	‘minister’ - 0.2220

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.

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

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 75% 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. Hence, punctuations, numbers and stopwords were thrown away and whole news were lowercased. Then remaining words were lemmatized by WordNetLemmatizer of nltk library. These processes decreased the number of the words.

Then TfIdf vectorizer was applied to the content with 4 and (1, 2) & (1, 3) for min_df and ngram_range parameters respectively. Then an article was represented with 3390 dimensional vector.

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

4. 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".

a. What did work?

For classical approaches, GridSearchCV method of scikit-learn 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 unbalanced 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 unbalanced dataset.

The best parameters for LogisticRegression() were;

```
{'C': 59.94842, 'class_weight': {1: 3, 0: 0.5}, 'penalty': 'l2'}
```

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	0.92	0.97	0.94	146
1.0	0.80	0.83	0.81	47
Accuracy			0.91	193
Macro avg.	0.87	0.88	**0.88**	193
Weighted avg.	0.91	0.91	0.91	193

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

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

```
{C': 1.0, 'class_weight': {1: 3, 0: 0.5}, 'kernel': 'linear'
```

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	0.94	0.92	0.94	144
1.0	0.78	0.82	0.80	49
Accuracy			0.90	193
Macro avg.	0.86	0.87	**0.87**	193
Weighted avg.	0.90	0.91	0.90	193

Table 4: Classification report of SVC on test set.

The results of the SVC is almost same with logistic regression.

To improve the reducible error due to the unbalanced data, data was under-sampled. New data had 146 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:

```
{'C': 1.0, 'class_weight': {1: 1, 0: 1}, 'penalty': 'l2'}
```

As it can be seen, class weights are equal and 1. 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	0.96	0.90	0.92	144
1.0	0.74	0.88	0.80	49
Accuracy			0.89	193
Macro avg.	0.85	0.89	**0.86**	193
Weighted avg.	0.90	0.89	0.89	193

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:

```
{'C': 1.0, 'class_weight': {1: 2, 0: 1}, 'kernel': 'linear'}
```

SVC model is still giving “class 1” more importance. 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	0.96	0.85	0.90	144
1.0	0.68	0.90	0.77	49
Accuracy			0.87	193
Macro avg.	0.82	0.88	**0.84**	193
Weighted avg.	0.89	0.87	0.87	193

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

After all, LogisticRegression is doing slightly better than SVC and models that are trained on original data are giving better results compared to the models that trained on unbalanced data. For the classical approaches, LogisticRegression is the selected model.

b. 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 606 equations. A simple perceptron with sigmoid activation gave the result on the **Table 7.**

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

Results are not even close to classical approaches. 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.

And finally 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.

5. 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 unbalanced 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.

REFERENCES

- Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. (2013). *An introduction to statistical learning: with applications in R*. New York :Springer,