Hyperpartisan News Detection
by using machine learning techniques

Scholtens, Richard s2956586
Wang, Remy S2212781
Wiltvank, Gijs s2749645

Abstract

This research set out to create a system which can determine if a news article is hyperpartisan or not by using binary classification. The research also used multi-classification to state if a news article is *left*, *left-center*, *least*, *right-center* or *right-center* or in political perspective, to state how biased an article is. This system will help the public in their evaluation of news articles to state the legitimacy of an article. To perform the classification tasks, this research used the Logistic Regression classifier and BERT Transformer model. The Multinomial Naive Bayes classifier is used as baseline and provided an F1-score of 81% for the binary, 59% for the multi-classification and 57% for combined classification. The Logistic Regression classifier had an F1-score of 92% and the BERT Transformer model an F1-score of 84% for the binary classification. The multi-classification resulted in an F1-score of 84% for the Logistic Regression classifier and 74% for the BERT Transformer model. The combined classification resulted in an F1-score of 82% for the Logistic Regression classifier.

1 Introduction

A shared task has been set out by PAN[1] for the SemEval 2019[2] congress. The Hyperpartisan News Detection (HND) task tries to fulfil the public need to evaluate articles. By doing so the public can be warned beforehand about the legitimacy of an article, which in turn leads to a better evaluation of the information presented. Present times call for a system to overcome unwanted influences which can lead to devastating problems in society so the public can be protected. The goal of this task is to build a system which allows for the classification of articles based on whether it is hyperpartisan and its bias. Classifying hyperpartisans for articles will be a binary classification task while classifying bias for articles will be a multi classification task. This research set out to fulfil both tasks by using machine learning techniques and furthermore it evaluates the combination of the classifications. Trying different classifiers will be useful for trying binary classification on the hyperpartisan labels and multi-classification on the bias labels. However, Support Vector Machine (SVM), Random Forest and Linear Regression classifiers did not show promising results. Therefore, this research uses the provided Multinomial Naive Bayes classifier as baseline and will only focus on Logistic Regression classifier and Transformer model using BERT as pre-trained language model.

2 Data

To train machine learning models input data must be gathered. The data is given by PAN for the SemEval 2019 congress. They released a portion of the released training/development set specific for HND. This portion comes from the total 1 million articles which has been split in training (200,000 left, 400,000 least, 200,000 right) and validation (50,000 left, 100,000 least, 50,000 right), where no publisher that occurs in the training set also occurs in the validation set. The overall bias of the publisher is used as labels for all articles. These labels are provided by BuzzFeed[3] journalists or MediaBiasFactCheck.com[4]. One must note that the trial data include characters which are replaced by question marks due to an error code. In this research a data set of 50000 articles, balanced according to being hyperpartisan or not, is used. This file is distributed as a csv

[1] https://pan.webis.de
[2] https://pan.webis.de/semeval19/semeval19-web/
[3] https://www.buzzfeed.com/
[4] https://mediabiasfactcheck.com/

file. One must consider that the instances come from a limited number of resources which could be biased in themselves. Publishers represented within the training data will not be represented within the test data to prevent that this information can be leveraged. Tables 1 and 2 represent how the text file is presented. Every line represents an instance and its labels are structured in the following order: id, hyperp, bias, url, labelby, publisher, date, title, text, and raw text. Table 3 shows the distribution for hyperpartisan per article. Table 4 shows the distribution for bias per article.

Table 1: First part of sample data

id	hyperp	bias	url	labeledby	publisher
1271352	False	least	https://reuters.com/article/...	publisher	https://reuters.com

Table 2: Second part of sample data

date	title	text	raw_text
2018-01-24	German stocks...	BERLIN/FRANKFURT, Jan...	b'<article id='1271352'...

Table 3: Distribution of hyperpartisan labels binary classification

Class	**Support**
True	24401
False	24066

Table 4: Distribution of bias labels multi-classification

Class	**Support**
Left	11659
Left-center	12272
Least	11361
Right-center	433
Right	12742

2.1 Cross-validation

To validate our classifier the K-fold method provided by Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, et al. (2011) will be used. By doing so the system splits the data into multiple training and test sets. This research folded the data five times allowing the system to use all data for validation and therefore, prediction. By performing the validation procedure five times on multiple sets more metric information is retrieved. Instead of using one metric to base the arguments of this research on the research will now have five metrics which in turn will help prevent bias. By using this method, it easier to draw important conclusions on both our algorithm and our data but also will provide the tools to develop a more robust system. However, one must note that the BERT Transformer model did not make use of this validation due to limited computational resources.

3 Method

Initially, we tried several different common classification algorithms in order to see which would fit the task best. The provided baseline was a Naive Bayes classifier, so we started there. The results of this classifier were not good, so we began experimenting with other algorithms. We tried a random forest classifier as well as an SVM but their results on the development data showed no real promise, thus we did not pursue them any further. Therefore, their results are not mentioned in this paper either. Furthermore, we experimented with logistic regression and got promising results on the development data and because of this, we decided to further develop this system.

3.1 Classification with logistic regression

The baseline provided to us used a simple count vectorizer to vectorize the data. We experimented with both this count vectorizer as well as a term frequency-inverse document frequency (tf-idf) vectorizer. In addition to that, we also tried a union of these two vectorizers, but this yielded no increase in the system's performance. As a result, we settled on using the count vectorizer with a n-gram range of one to three meaning unigrams, bigrams and trigrams. This vectorizer also makes use of a list of English stop words to filter out of the data. The used list of stop words is a standard list provided by the NLTK library developed by Loper and Bird (2002). For tuning the model, we wanted to use grid search developed by Pedregosa et al. (2011). However, we found that this was computationally too challenging and would take up too much of our time. As a result, there is experimented with the different parameters of the logistic regression model. Specifically, the 'penalty' parameter and the 'solver' parameter. The final model makes use of the default 'liblinear' solver and uses the 'l2' penalty. Furthermore, the 'multi-class' parameter is set to 'auto'.

3.2 Filtering impossible prediction combinations

During the development process, we had a different idea which was to try and prevent impossible predictions. For example, if an article is predicted to be hyperpartisan, its bias can only be *left* or *right* out of all the possible biases (*left*, *right*, *left-center*, *right-center*, *least*). Therefore, we wanted to see if we could make sure that the system only predicts possible labels. The method we tried consisted of splitting the five-way classification task into a binary and a three-way task. For both tasks, a model was fitted on the data relevant to that task. These two separate data sets were created based on the *True*, *False* values in the development set. These two models are then used to predict on two test sets which are created by splitting the given test set into two sets relevant for each task. This is done by using the predictions of the earlier model on the test set to determine to which task a given article belongs. These predictions are then combined to be the final predictions.

3.3 Transformers library by Huggingface

Transformer models have been making strides in the field of Natural Language Processing (Devlin, Chang, Lee, and Toutanova, 2018). Because of the multiple pre-trained Transformers available, it is easy to take advantage of these models that have been trained using high computational power. For this task we have used the Transformers library by Huggingface to initialize a model, train and evaluate the data set [5]. For the pre-trained model we use the bert-based-cased, which was trained using 12-layer, 768-hidden, 12-heads, 110M parameters on cased English text. First, we load the data in the correct form using pandas data frames. Second, we used the bert-base-cased as model and specified the parameters. We mostly used the default parameters due to the lack of computational resources. After these steps we can train and evaluate our model. Finally, we can use the model to make predictions on unseen data.

4 Results

In this section the results of the experiments are presented in tables. Tables 5 and 6 show the results for the baseline, 7 and 8 for logistic regression without filtering of impossible combinations, 9 and 10 for logistic regression that does filter impossible combinations. Tables 11 and 12 show the results for the Transformer model using the bert-base-cased pre-trained language model.

[5] https://github.com/huggingface/transformers

Table 5: metrics per label of the baseline model on the development set (k=5)

Label	**F1 score**	**Accuracy**	**Precision**	**Recall**
True	0.80	0.80	0.82	0.78
False	**0.81**	**0.81**	0.79	**0.83**
Left	0.71	0.71	0.66	0.78
Left-center	0.67	0.67	0.64	0.70
Least	0.68	0.68	0.72	0.63
Right-center	0.06	0.06	**1**	0.03
Right	0.78	0.78	0.82	0.72
True left	0.69	0.69	0.71	0.69
True right	0.77	0.77	0.85	0.69
False left-center	0.66	0.66	0.66	0.67
False least	0.68	0.68	0.74	0.64
False right-center	0.06	0.06	1	0.03

Table 6: Macro F1 score and accuracy of the baseline on the development set (k=5) per task

Task	**F1 score**	**Accuracy**
Binary	**0.81**	0.81
Multi-class	0.59	0.71
Both labels	0.57	0.67

Table 7: metrics per label of the simple logistic regression model on the development set (k=5)

Label	**F1 score**	**Accuracy**	**Precision**	**Recall**
True	0.92	0.92	0.91	0.93
False	0.92	0.92	0.93	**0.91**
Left	0.84	0.84	0.82	0.87
Left-center	0.85	0.85	0.82	0.88
Least	0.91	0.91	0.92	0.90
Right-center	0.53	0.53	0.37	0.53
Right	**0.94**	0.94	0.98	0.91
True left	0.83	0.83	0.84	0.83
True right	0.94	**0.94**	**0.98**	0.90
False left-center	0.85	0.85	0.84	0.86
False least	0.91	0.91	0.93	0.89
False right-center	0.49	0.49	0.95	0.33

Table 8: Macro F1 score and accuracy of the simple logistic regression model on the development set (k=5) per task

Task	**F1 score**	**Accuracy**
Binary	**0.92**	**0.92**
Multi-class	0.84	0.89
Both labels	0.82	0.87

Table 9: metrics per label of the complex logistic regression model on the development set (k=5)

Label	F1 score	Accuracy	Precision	Recall
True	0.92	**0.92**	0.90	**0.93**
False	0.92	0.92	0.93	0.90
Left	0.82	0.82	0.81	0.84
Left-center	0.83	0.83	0.80	0.87
Least	0.88	0.88	0.89	0.88
Right-center	0.61	0.61	**1.00**	0.44
Right	**0.94**	0.94	0.97	0.90
True left	0.76	0.75	0.74	0.77
True right	0.85	0.85	0.90	0.81
False left-center	0.75	0.75	0.70	0.80
False least	0.81	0.81	0.81	0.82
False right-center	0.64	0.64	0.96	0.48

Table 10: Macro F1 score and accuracy of the complex logistic regression model on the development set (k=5) per task

Task	F1 score	Accuracy
Binary	**0.92**	**0.92**
Multi-class	0.82	0.87
Combined labels	0.76	0.80

Table 11: metrics per label of the Transformer model on the development set

Label	F1 score	Accuracy	Precision	Recall
True	**0.84**	**0.84**	0.82	**0.87**
False	0.83	0.83	0.86	0.80
Left	0.75	0.75	0.72	0.78
Left-center	0.68	0.68	0.69	0.68
Least	0.75	0.75	0.74	0.76
Right-center	0.02	0.02	**1.00**	0.01
Right	0.78	0.78	0.80	0.76

Table 12: Macro F1 score and accuracy of the Transformer model on the development set

Task	F1 score	Accuracy
Binary	**0.84**	**0.84**
Multi-class	0.74	0.73

5 Discussion

This section will discuss the results presented in the previous section. By comparing the baseline, logistic regression classifiers and BERT Transformer model it is possible to derive conclusions about the performed experiments.

5.1 Baseline

As can be seen in table 5 and 6 the baseline scored on the binary classification an F1-score of 81%, multi-classification 59% and on the combined labels 57%. There is a substantial difference between binary and multi-classification. However, one must conclude the binary classification itself is an easier task to perform. The multi-classification falls behind when compared to the binary classification. When evaluating the combined labels, it performs worse than the separate classification. This is due to the fact that this evaluation method is more strict.

5.2 Logistic Regression without filtering

Tables 7 and 8 show the results of the simple logistic regression model. As can be seen, this model achieves higher F1-scores and accuracy scores when compared to the baseline. On the binary task it achieves an F1-score of 92%, which is around 11% higher than the baseline. The biggest increase in performance is achieved in the multi-class classification. The logistic regression model achieves an F1-score of 84% in comparison to the 59% of the baseline, which is an increase of 25%. The performance drops equally to the baseline when evaluating on the combined labels, around 2%. Therefore, one must conclude that the logistic regression method outperforms the baseline.

5.3 Logistic regression with filtering

The results of the logistic regression model that filters out impossible combinations can be seen in tables 9 and 10. The F1-score on the binary task remains unchanged at 92%. The performance on the multi-class drops when compared to the simple logistic regression model. The F1-score is now 82% which is around 2% lower. One can conclude that while this system cannot predict impossible combinations, it performs worse. This could be because the binary predictions are not 100% accurate and therefore automatically introduce some error in the multi-class predictions. This could be the reason that this model is outperformed by the simpler model.

5.4 Transformer using BERT model

The results of the Transformer using BERT model can be seen in tables 11 and 12. The F1-score for the binary task is 84% while the performance for the multi-class is 74%. The Transformer model outperforms the baseline by 3% on the binary score and around 15% for the multi-class. Due to complications of running the Transformer model on Google Colab, time constraints and the lack of promising results, the joint label task was not performed, and thus reported, for the Transformer model. It can be concluded that the logistic regression model performs better on both tasks than the Transformer model, but the Transformer model does outperform the baseline, as seen in the results. For future work, other pre-trained models can be used to see if this increases the performance such as XLNet, XLM and RoBERTa. The Transformer model mostly used the default parameters with an epoch of only 1 or 2 due to the lack of computational resources. Increasing the amount of epochs could improve the performance. Acquiring a GPU that you can use could combat this problem.

6 Division of labour

The initial coding was done by Richard. This includes file handling, grid search and a k-fold function. Furthermore, he implemented a few classification algorithms in order to see which would show the most promise. Further experimentation with additional classifiers as well as the tuning of the final logistic regression model was done by Gijs. He also wrote the script that prints all the metrics and writes the predictions to a .txt file. Remy implemented a script that makes use of the BERT transformer model.

Richard wrote the abstract, introduction and the data sections. Gijs wrote the method for Logistic Regression and reported the results. Remy wrote the method for the BERT transformer, conclusion /discussion. Richard and Gijs also performed the final spelling and grammar check and wrote additional introduction parts.

References

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Loper, E. and S. Bird (2002). Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics.*

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research 12*(Oct), 2825–2830.