The Sally Smedley Hyperpartisan News Detector at SemEval-2019 Task 4: Learning Classifiers with Feature Combinations and Ensembling

Kazuaki Hanawa*[1,2], Shota Sasaki*[1,2], Hiroki Ouchi[1,2], Jun Suzuki[2,1], Kentaro Inui[2,1]
(* equal contribution)
[1]RIKEN AIP, [2]Tohoku University
{hanawa, sasaki.shota, hiroki.ouchi, jun.suzuki, inui}
@ecei.tohoku.ac.jp

Abstract

This paper describes our system submitted to the formal run of SemEval-2019 Task 4: Hyperpartisan news detection. Our system is based on a linear classifier using several features, i.e., 1) embedding features based on the pre-trained BERT embeddings, 2) article length features, and 3) embedding features of informative phrases extracted from the by-publisher dataset. Our system achieved 80.9% accuracy on the test set for the formal run and got the 3rd place out of 42 teams.

1 Introduction

Hyperpartisan news detection (Kiesel et al., 2019; Potthast et al., 2018) is a binary classification task, in which given a news article text, systems have to decide whether or not it follows a hyperpartisan argumentation, i.e., "whether it exhibits blind, prejudiced, or unreasoning allegiance to one party, faction, cause, or person" (2019). As resources for building such a system, the `by-publisher` and `by-article` datasets are provided by the organizer. The `by-publisher` dataset is a collection of news articles labeled with the overall bias of the publisher as provided by BuzzFeed journalists or MediaBiasFactCheck.com. The `by-article` dataset is a collection labeled through crowdsourcing on an article basis. This data contains only the articles whose labels are agreed by all the crowd-workers. The performance measure is accuracy on a balanced set of articles.

Our system is based on a linear classifier using several types of features mainly consisting of 1) embedding features based on the pre-trained BERT embeddings (Devlin et al., 2018) and 2) article length features and 3) embedding features of informative phrases extracted from `by-publisher` dataset. Our system achieved 80.9% accuracy on the test set for the formal run and got 3rd place out of 42 teams in the formal run.

2 System Description

This section first presents an overview of our system and then elaborate on the feature set.

2.1 Overview of Our System

Our system is based on a linear classifier that models the conditional probability distribution over the two labels (positive or negative) given features. Let $\mathbf{f}$ be a feature vector. $\mathbf{W}$ denotes a trainable weight matrix, and $\mathbf{b}$ is a trainable bias vector, where $\mathbf{f} \in \mathbb{R}^D$, $\mathbf{W} \in \mathbb{R}^{D\times 2}$ and $\mathbf{b} \in \mathbb{R}^2$, respectively. Then, we compute the conditional probability as follows:

$$\mathbf{y} = \mathtt{softmax}(\mathbf{W}^\top \mathbf{f} + \mathbf{b}). \quad (1)$$

where, $\mathtt{softmax}(\cdot)$ represent the softmax function that receives an N-dimensional vector $\mathbf{x}$ and returns another N dimensional vector, namely:

$$\mathtt{softmax}(\mathbf{x}) = \frac{\exp(\mathbf{x})}{\sum_i \exp(x_i)}, \quad (2)$$

and $\mathbf{x} = (x_1, \ldots, x_N)^\top$. After the softmax computation, we obtain the two-dimensional vector $\mathbf{y} \in \mathbb{R}^2$. We assume that the first dimension of this vector represents the probability of the positive label, and the second one represents that of the negative label.

To boost the performance, we concatenate three types of features, $\mathbf{f}_1$, $\mathbf{f}_2$, and $\mathbf{f}_3$, into the single feature vector $\mathbf{f}$, where $\mathbf{f}_1 \in \mathbb{R}^{D_1}$, $\mathbf{f}_2 \in \mathbb{R}^{D_2}$ and $\mathbf{f}_3 \in \mathbb{R}^{D_3}$ and $D = D_1 + D_2 + D_3$.

As $\mathbf{f}_1$, $\mathbf{f}_2$ and $\mathbf{f}_3$, we design the following features.

- $\mathbf{f}_1$: BERT feature (Section 2.2)

- $\mathbf{f}_2$: Article length feature (Section 2.3)
- $\mathbf{f}_3$: Informative phrase feature (Section 2.4)

For training our classifiers, we used only the `by-article` dataset but not the `by-publisher` dataset. This is because the labels of the `by-publisher` dataset turned out rather noisy. In our preliminary experiments, we found that the performance drops when training the classifiers on the `by-publisher` dataset.

Furthermore, we apply the following three techniques.

1. **Word dropout**: We adopted word dropout (Iyyer et al., 2015) for regularization. The dropout rate was set to 0.3.
2. **Over sampling**: As mentioned above, the gold label distribution of the training set is unbalanced while that of the test set is balanced. We deal with this imbalance problem by sampling 169 (407 − 238) extra examples from hyperpartisan data.
3. **Ensemble**: We trained 100 models with different random seeds. In addition, we trained models for 40, 50, 60 and 70 epochs for each seed. Consequently, we finally average the output of these 400 models.

2.2 BERT Feature

Our system uses BERT (Devlin et al., 2018). As a strategy for applying BERT to downstream tasks, Devlin et al. (2018) recommends a fine-tuning approach, which fine-tunes the parameters of BERT on a target task. In contrast, we adopt a feature-based approach, which uses the hidden states of the pre-trained BERT in a task-specific model as input representation. A main advantage of this approach is computational efficiency. We do not have to update any parameters of BERT. Once we calculate a fixed feature vector for an article, we can reuse it across all the models for ensemble.

In our system, we used BERT to compute a feature vector $\mathbf{f}_1$ for an input article. Specifically, we first fed an article to the pre-trained BERT model as input and extracted the representations of all the words from the top four hidden layers. Then, to summarize these representations into a single feature vector $\mathbf{f}_1$, we tried the following three methods.

1. **Average**: Averaging the representations of all the words in an article.
2. **BiLSTM**: Using the representations as input to BiLSTM. This is the same method as the best performing one reported by Devlin et al. (2018).
3. **CNN**: Using the representations as input to CNN in the same way as Kim (2014).

As we describe in Section 3.2, we finally adopted the averaged BERT vectors as $\mathbf{f}_1$.

2.3 Article Length Feature

As $\mathbf{f}_2$, we design a feature vector representing the length of an input article. In our preliminary experiments, we found a length bias in hyperpartisan articles and non-hyperpartisan articles. Thus a vector representing the length bias is expected to be useful for discriminating these two types of articles.

Specifically, we define a one hot feature vector $\mathbf{f}_2$ representing distribution of the lengths of articles (the number of words in an article). To represent the length of an article as a vector, we make use of histogram bins. Consider the 100-ranged histogram bins. The first bin represents the length 1 to 100, and the second one represents the length 101 to 200. If the length of an article is 255, the value of the third bin (201 to 300) takes 1. In the same way, the third element of the length vector takes 1 and the others 0, i.e., $\mathbf{f}_2 = [0, 0, 1, 0, 0, \cdots]$. In our system, we set the dimension of the vector as $D_2 = 11$, whose last (11-th) element corresponds to the length longer than 1000.

2.4 Informative Phrase Feature

In the development set, we found some phrases informative and useful for discriminating whether or not a given article is hyperpartisan. We extracted such informative phrases and mapped them to a feature vector $\mathbf{f}_3$. In this section, we first explain the procedure of extracting informative phrases, and then we describe how to map them to a feature vector.

2.4.1 Phrase Set Creation

To create an informative phrase set, we exploit the `by-publisher` articles. Basically, we take advantage of chi-squared statistics of N-grams ($N = 1, 2, 3$).

Creation of S_h First, we calculate each chi-squared value χ_{x_i} of N-gram x_i appearing in the `by-publisher` articles as follows:

$$\chi_{x_i} = \frac{(O - E)^2}{E}. \quad (3)$$

O and E are defined as follows:

$$O = f_{\text{false}}(x_i), \quad (4)$$

$$E = \frac{T_{\text{false}} \times f_{\text{true}}(x_i)}{T_{\text{true}}}, \quad (5)$$

where $f_{\text{true}}(\cdot)$ and $f_{\text{false}}(\cdot)$ are functions that calculate the frequency of x_i in hyperpartisan articles and non-hyperpartisan articles, respectively. T_{true} and T_{false} are the summation of the frequency of all N-grams in hyperpartisan articles and non-hyperpartisan articles, respectively.

Then, based on the chi-squared values χ_{x_i}, we sort and select top-M N-grams. We can obtain a typical N-gram set (hereinafter, referred to as S_h) that is informative for judging whether an article is hyperpartisan or not.[1] In our system, we use $M = 200,000$.

S_h can mostly catch the characteristics of hyperpartisan articles. However, S_h may include some N-grams that are typical of a certain publisher. This is because the `by-publisher` dataset is labeled by the overall bias of the publisher as provided by BuzzFeed journalists or MediaBiasFactCheck.com.

Creation of S_p To remedy this problem, we create another phrase set S_p consisting of N-grams that are typical of a publisher, and exclude them from S_h.

Here we consider a certain publisher p_l. First, we calculate each chi-squared value of N-gram x_i in the same way as Eq 3,4,5 for S_h, but here we consider $true$ and $false$ as appearing in articles of publisher p_l and articles of other publishers, respectively, instead of appearing in hyperpartisan articles and non-hyperpartisan articles. Then, we pick up only N-gram x_i where $f_{\text{true}}(x_i)$ is less than $f_{\text{false}}(x_i)$ and sort all of them by the chi-squared values. This is because we aim to exclude only N-grams that are typical of a certain publisher.

Next, we try four types of ways to create S_{p_l} from sorted χ_x value statistics. Here j denotes the index of the N-gram list sorted by χ_x values, i.e., χ_{x_1} is the highest value in all calculated χ_x values.

[1]Actually, we cut off N-gram x if $f_{\text{true}}(x)$ is more than 100,000 for S_h.

1. **Top-T_o**: The first setting is to select top-T_o N-grams. Concretely, S_{p_l} is defined as follows:

$$S_{p_l} = \{x_j | j \leq T_o\}. \quad (6)$$

2. **χ-based**: The second setting is to select N-grams based on χ values. Concretely, S_{p_l} is defined as follows:

$$S_{p_l} = \{x_j | \chi_{x_j} > T_c\}. \quad (7)$$

3. **f_{true}-based**: The third setting is to select N-grams based on $f_{\text{true}}(x_j)$ values. Concretely, S_{p_l} is defined as follows:

$$S_{p_l} = \{x_j | f_{\text{true}}(x_j) > T_f, j \leq T_m\}. \quad (8)$$

4. **f_{true}-f_{false} ratio-based**: The fourth setting is to select N-grams based on ratios between f_{true} and f_{false}. Concretely, S_{p_l} is defined as follows:

$$S_{p_l} = \left\{ x_j \,\middle|\, \frac{f_{\text{true}}(x_j)}{f_{\text{false}}(x_j)} > T_r, j \leq T_m \right\}. \quad (9)$$

T_o, T_c, T_f, T_r and T_m are hyper-parameters[2].

Next, we obtain S_p defined as follows:

$$S_p = \bigcup_l S_{p_l}. \quad (10)$$

At last, we obtain an filtered N-gram set S defined as follows:

$$S = S_h \setminus S_p. \quad (11)$$

2.4.2 Phrase Embedding

We map each of the obtained N-gram phrase set S to a feature vector $\mathbf{f}_3$. We exploited GloVe vectors (Pennington et al., 2014) instead of one-hot vectors in order to facilitate generalization.

First, we enumerate N-grams included in an article and compute each N-gram vector. Each vector is the average of GloVe vectors of included words. For example, the vector for the phrase "news article" is computed as follows:

$$\frac{\texttt{GloVe}(\text{news}) + \texttt{GloVe}(\text{article})}{2}.$$

Here, $\texttt{GloVe}(w)$ denotes the GloVe (`glove.840B`[3]) vector of the word w. Then, we compute $\mathbf{f}_3$ as the average of all N-gram vectors included in the article.

[2]In our experiments, we fix T_m to 200,000.

[3]https://nlp.stanford.edu/projects/glove/

Method	Accuracy
Average	0.760
BiLSTM	0.712
CNN	0.758

Table 1: Accuracy of hyperpartisan classification for each operation on BERT vectors.

3 Experiments

3.1 Settings

We trained linear classifiers on the `by-article` dataset (not on the `by-publisher` dataset). In order to estimate the performance in the test set with each setting, we conducted 5-fold cross validation on `by-article` dataset. For optimization of the classifiers, we use Adam with learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$. We set the minibatch size to 32. Note that we did not take ensemble approach in the experiments we report in Section 3.2 and Section 3.3 for efficiency.

3.2 Operation on BERT Vectors

We conducted experiments on each of the three methods for BERT vectors (`BERT-Base, Uncased`[4]) mentioned in Section 2.2. In this experiment, we only used $\mathbf{f}_1$ as a feature vector $\mathbf{f}$, i.e., without using $\mathbf{f}_2$ and $\mathbf{f}_3$.

Table 1 shows the performance in each setting. The averaging method was the best performance this time. We therefore decided to adopt average BERT vectors as $\mathbf{f}_1$ for the evaluation of the formal run. In addition, we also used averaged BERT vectors as $\mathbf{f_1}$ in the following experiments.

3.3 Method to Create N-gram Set

As mentioned in Section 2.4, we examined which method is the best to create an informative N-gram set S (and $\mathbf{f}_3$ derived from them). In this experiment, we also used $\mathbf{f}_1$ and $\mathbf{f}_2$ with $\mathbf{f}_3$ as a feature.

Table 2 shows the performance in each setting. The performance was the best when we adopted Top-T_o ($T_o = 100$) for S_p creation. We therefore used $\mathbf{f}_3$ created in this setting.

3.4 Ablation

To verify the contribution of each three types of features, we conducted feature ablation experiments. In addition, we investigated to what extent ensemble approach improve the performance. In this experiments, we use only 10 (not 100) different random seeds for ensemble due to time constraints.

[4]https://github.com/google-research/bert

Method	Accuracy
Top-T_o ($T_o = 100$)	0.777
Top-T_o ($T_o = 1000$)	0.771
χ-based ($T_c = 20000$)	0.752
f_{true}-based ($T_f = 50$)	0.754
f_{true}-based ($T_f = 150$)	0.764
f_{true}-f_{false} ratio-based ($T_r = 0.5$)	0.754
f_{true}-f_{false} ratio-based ($T_r = 0.8$)	0.771
f_{true}-f_{false} ratio-based ($T_r = 1.0$)	0.756

Table 2: Accuracy of hyperpartisan classification for each method to create N-gram set.

Features	Ensemble	Accuracy
$\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}$	true	0.788
$\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}$	false	0.777
$\mathbf{f_1}, \mathbf{f_2}$	false	0.769
$\mathbf{f_1}$	false	0.760

Table 3: Result of ablation.

Table 3 shows the performance in each setting. We found that $\mathbf{f_2}$ and $\mathbf{f_3}$ improved the accuracy by about 0.01, respectively. Additionally, by using the ensemble method, the accuracy increased by about 0.01.

4 Conclusion

We described our system submitted to the formal run of SemEval-2019 Task 4: Hyperpartisan news detection. We trained a linear classifier using several features mainly consisting of 1) BERT embedding features, 2) article length features indicating the distribution of lengths of articles and 3) embedding features derived from filtered N-grams that are typically found in hyperpartisan articles. Our system achieved 80.9% accuracy on the test set for the formal run and got 3rd place out of 42 teams.

Acknowledgments

This work was supported by JST CREST Grant Number JPMJCR1301, Japan.

References

2019. Pan @ SemEval 2019 - Hyperpartisan News Detection. https://pan.webis.de/semeval19/semeval19-web/index.html.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1681–1691.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. Semeval-2019 task 4: Hyperpartisan news detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 231–240.