Team *Peter Brinkmann* at SemEval-2019 Task 4: Detecting Biased News Articles Using Convolutional Neural Networks

Michael Färber
University of Freiburg, Germany
michael.faerber@cs.uni-freiburg.de

Agon Qurdina
University of Prishtina, Kosovo
agon.qurdina@studentet.uni-pr.edu

Lule Ahmedi
University of Prishtina, Kosovo
lule.ahmedi@uni-pr.edu

Abstract

In this paper, we present an approach for classifying news articles as *biased* (i.e., *hyperpartisan*) or *unbiased*, based on a convolutional neural network. We experiment with various embedding methods (pretrained and trained on the training dataset) and variations of the convolutional neural network architecture and compare the results. When evaluating our best performing approach on the actual test data set of the SemEval 2019 Task 4, we obtained relatively low precision and accuracy values, while gaining the highest recall rate among all 42 participating teams.

1 Introduction

Hyperpartisan news detection describes the task of given a news article text, decide whether it follows a hyperpartisan argumentation, i.e., whether it exhibits blind, prejudiced, or unreasoning allegiance to one party, faction, cause, or person (Kiesel et al., 2019). In recent years, hyperpartisan news detection, which we consider synonymous to news bias detection, has attracted the interest of researchers and various approaches for news bias detection have been developed (Recasens et al., 2013; Baumer et al., 2015; Baly et al., 2018). However, the definition of *bias* and the task set-up of identifying biased news articles differs from authors to authors. For instance, authors might consider the bias in terms of the writing style, while others might consider it in relation to fact selection (Hamborg et al., 2018). In this paper, we use the definition and data set of SemEval 2019 Task 4 (Kiesel et al., 2019), which deliberately uses the generic definition outlined at the beginning. Note that news bias detection differs from related tasks such as opinion finding (Ounis et al., 2006; Macdonald et al., 2007), sentiment analysis, fake news detection (Potthast et al., 2018), claim assessment (Popat et al., 2016), argumentation mining on news articles (Palau and Moens, 2009), and personality detection based on texts (Mairesse et al., 2007).

From a technical perspective, in recent years deep learning techniques have outperformed traditional methods concerning various NLP tasks. This also applies to news article classification tasks (Ounis et al., 2006; Macdonald et al., 2007). Indeed, in the SemEval Twitter sentiment analysis competition in 2015, 2016, and 2017 (Rosenthal et al., 2015; Nakov et al., 2016; Rosenthal et al., 2017), among the most popular (and apparently effective) deep learning techniques were convolutional neural networks (CNNs). Thus, we decided to build a hyperpartisan news classifier based on a CNN. Next to our basic model, we also develop and evaluate variations of our model.

2 Approach

In the following, we outline our approach for news bias detection.[1]

2.1 Preprocessing

Given a set of news articles as input, we preprocessed them along the following steps:

Text Cleaning. We replaced the new lines by spaces, expanded contractions, and removed stop words, HTML tags, and special characters from the articles' content.

Texts to Sequences. The articles' content was tokenized and a word dictionary (size: 1,207,438) was generated.

Sequence Padding. We applied a padding with

[1]Note that our team's name is dedicated to *Peter Brinkmann*, the German journalist who asked the question that ultimately fractured the Berlin wall in 1989. The source code of the implementation is available online at https://github.com/michaelfaerber/SemEval2019-Task4.

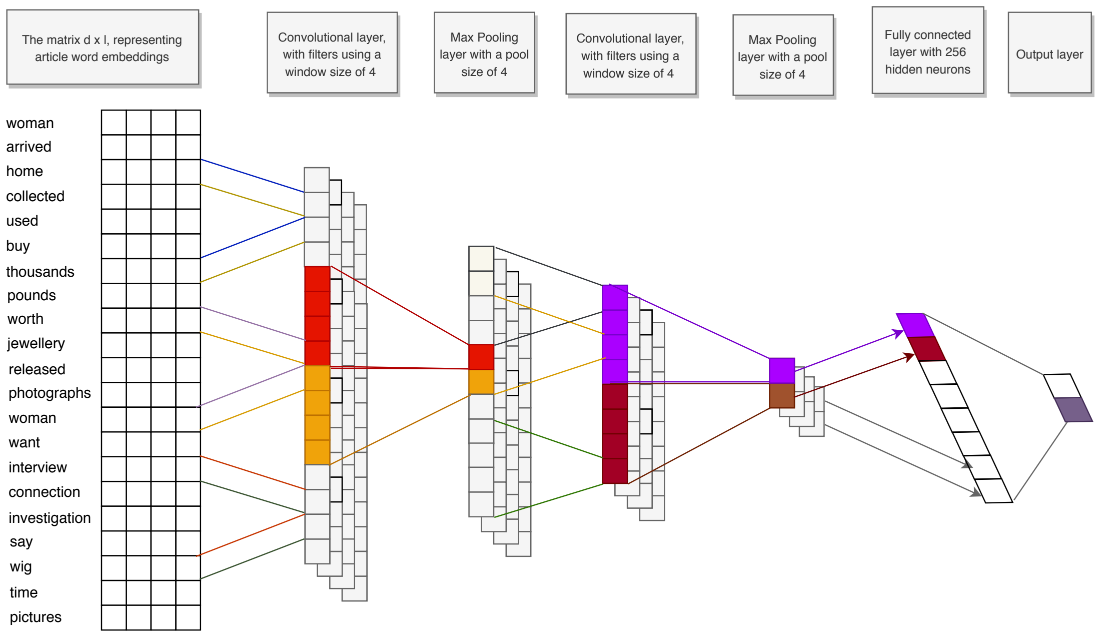
The matrix d x l, representing article word embeddings
Convolutional layer, with filters using a window size of 4
Max Pooling layer with a pool size of 4
Convolutional layer, with filters using a window size of 4
Max Pooling layer with a pool size of 4
Fully connected layer with 256 hidden neurons
Output layer
woman
arrived
home
collected
used
buy
thousands
pounds
worth
jewellery
released
photographs
woman
want
interview
connection
investigation
say
wig
time
pictures

Figure 1: Architecture of our system used for classifying sentences.

a fixed sequence length l. We set $l = 5000$. Thus, around 0.04% of the sequences were truncated.

2.2 Basic Model Architecture

Our basic architecture is shown in the Figure 1. We use a CNN architecture that is based on Kim et al.'s approach for sentence classification (Kim, 2014). His proposed architecture has been widely applied for various tasks in the past. The CNN consists of two subsequent one-dimensional convolutional neural networks layers, appended by MaxPooling layers, and a dense neural networks layer processing the output of the second CNN layer. The model is completed by a final output layer that uses the sigmoid activation function to return a binary output (i.e., the classification into *biased* or *non-biased*).

In the following, we describe the architecture in more detail:

Input layer. Considering article's words were embedded into d-dimensional vectors, the final matrix used as input to the model can be written as $I = l \times d$ where l is the chosen sequence length (i.e., the length of the articles). Recall that $l = 5000$ in our setting.

First convolutional layer. The first transformation this embedded input goes through is a convolutional layer with f 1-dimensional filters of length k. Thus, the layer weights can be considered a matrix of shape $Wc \in R^{f \times k}$.

We chose as filter size $f = 64$, while using a filter length of $k = 4$. In our context, having 1-dimensional filters means that for each word in an article, its three adjacent words are considered as the context of the word. The output of the convolutional layer then is $C = conv(I, Wc)$ where $conv$ is the convolutional operation applied to input I using the weights matrix Wc. This operation includes applying the *ReLu* activation function to complete weights calculations. Also, a dropout function is used to prevent overfitting. We used a dropout rate of 0.2, which means 20% of the weights, chosen randomly, during each training epoch are set to 0.

First max pooling layer. The above output C is then considered the input to a 1-dimensional Max Pooling layer. The purpose of the layer is to try extracting only the most important features of the convolution outputs. This is done by keeping only the max value from a pool size p. As we chose $p = 4$, the output of this layer can be written as $M = max\_pool(C, p)$. This operation reduces the number of weights by four times.

Second convolutional layer and max pooling layer. The output M of the max pooling layer is the input of the second convolutional layer, and the whole process described above is applied to this input, to get a final output M_2.

Fully connected layer. Given the two-dimensional matrix of weights from the last step, the next layer in the network is a fully connected one with a size of 256 hidden neurons. But in or-

Table 1: Distribution of biased and unbiased articles ("a." for articles) in the training and validation data set.

	Training	Test
Total # samples	588837	147210
# samples w/ $l > 2500$	0.62%	0.58%
# samples w/ $l > 5000$	0.04%	0.03%
# of biased articles	290513	72629
# non-biased articles	298324	74581
% biased articles	49.34	49.34
% of non-biased articles	50.66	50.66

Table 2: Hyperparameters.

Parameter	Value
CNN filter size for CNN	64
CNN kernel size for CNN	4
MaxPooling1D pool size	4
Dropout rate	0.2
Dense layer units	256
Layers Activation function	ReLu
Optimizer	Adam
Learning Rate	Adaptive (0.001→0.00001)
Loss function	Binary crossentropy
Batch Size	32

der for the convolutional output to serve as an input to this layer, it needs to be reduced in dimensionality. The way we chose to do that was using the flatten method, which keeps all of the values but flatten them in a long vector. A *ReLU* activation function and a dropout layer with a rate of 0.5 were used here.

Output layer. The last layer is a fully connected layer with one neuron. The sigmoid activation function is used to provide a binary output.

2.3 Architecture Variations

We developed and evaluated the following architecture variations:

1. The first variation only keeps the most important features by using the *GlobalMaxPooling* method. Keeping only the max values has shown good results when used with convolutional layers (Scherer et al., 2010).

2. The second variation uses the *flatten* method, which also transforms the convolutional filters weights to a one-dimensional vector, but does that by keeping all of the values (text features) from the convolutional filters and concatenates them in the resulting vector. Obviously, the length of the output from this method will be greater, usually much greater, than the previous method.

3 Evaluation

3.1 Data Set

Note that the actual SemEval 2019 Task 4 test data set is hidden and only used for submission. Thus, we used the training and validation data set of the SemEval 2019 Task 4 data set for training and testing our models before the SemEval submissions. Note also that, the biased news articles in the training and test data set always originate from sources other than the non-biased news articles. Biased news articles originated mainly from `foxbusiness.com`, `counterpunch.org`, `motherjones.com`, `truthdig.com`, and `dailywire.com`, while unbiased news articles originated mainly from `abqjournal.com` and `apnews.com`. The SemEval 2019 Task 4 is, thus, to some degree artificial, as, in reality, the source of a given article could be used as a feature for the classification.

Due to this correlation between the article's bias and its publisher, to have a generic model as much as possible it was important to have articles from the same publisher present in both sets. We decided to merge the training and validation data set, to shuffle the data randomly, and to split it into a training, validation, and testing data set by 60:20:20. Table 1 shows basic statistics about the used training data set and test data set.

3.2 Evaluation Settings

We developed our models using Keras v2.1.2 with a Tensorflow v1.0.0 backend. Training the model was performed on a machine with 64GB memory and a GeForce GTX 1080 Ti GPU.

We implemented and evaluated our basic model using several word-based embedding methods, where an embedding vector is generated for each unique word on the text corpus. These embeddings can be categorized into two main categories: (1) pretrained word vectors and (2) custom word vectors (here, trained on the articles' content).

We fine-tuned the hyperparameters of our basic model using the dedicated validation data set. In the end, we used the parameters as shown in Table 2. Note that these optimal hyperparameters showed to be the best-performing ones on both Google's prebuilt word2vec and the custom word2vec which we had trained on our training data.

Table 3: Evaluation results on the custom validation split using various embedding methods.

Embedding	# Dim.	Accuracy	Precision	Recall	F1
Google's word2vec (general)	300	0.9255	0.9295	0.9197	0.9246
Stanford's GloVe (general)	100	0.9198	0.9161	0.9216	0.9188
Facebook's fastText (general)	300	0.9277	0.9378	0.9021	0.9196
Custom word2vec	100	0.9234	0.9246	0.9197	0.9222
Custom fastText	100	0.9114	0.9309	0.8860	0.9079
Custom news word2vec	100	0.9123	0.9142	0.9073	0.9108

Table 4: Results for the adapted CNN architecture.

Architecture	Accuracy
Model using GlobalMaxPooling	0.9225
Model using Flatten	0.9255

3.3 Evaluation Results

3.3.1 Evaluating the Basic Architecture

Table 3 presents the evaluation results for all used embedding methods. The embedding models gave similar results, with some slight differences in the model accuracies for some of them. Thus, we decided to go with one of the better performing models for the final SemEval test runs (see Sec. 3.3.3).

Although the fastText word embeddings are said to perform as well as word2vec embeddings while being trained much faster (Grave et al., 2017) and although the domain-specific pre-trained word2vec embeddings are said to perform better than the general pre-trained word2vec embeddings (Kim, 2014), the general pre-trained word2vec embeddings lead to the best results in our evaluation. A reason for that could be that the words used in the news articles are more spread in terms of the domains to which they belong. Thus, the word vectors trained by Google on 100 billion words of Google News performed even better than a specific word2vec model trained on millions of news articles.

3.3.2 Evaluating the Architecture Variations

We trained the extended models with the same hyperparameters as our basic model and used Google's word2vec as word embedding method based on our results from Sec. 3.3.1. The evaluation results are shown in Table 4.

Even though the training times for the flatten method were much longer due to the huge difference in terms of the number of trainable parameters, we obtained slightly better results than the GlobalMaxPooling method. Thus, the model chosen for the SemEval submission was the model employing the flattening method.

3.3.3 Evaluating on SemEval's Test Data Set

Applying our basic model – using our second architecture variation, Google's word2vec embedding model, the hyperparameters shown in Table 2 and the model itself trained using our mixed, "artificial" data sets – on the official SemEval test data set via official approach submissions, we obtained accuracy of 0.602, a precision of 0.560, a recall of 0.955, and a F1 score of 0.706. It becomes apparent that the precision value is considerably low, while the recall rate is the highest achieved rate among all submitted systems. Based on our investigations so far, we believe that one reason for the low accuracy on the evaluation data set is the labeling of the used data sets: while the train and test data set's labels depend solely on the article's publisher, the labels of the evaluation data set were hand-labeled on a single article basis.

4 Conclusion

In this paper, we presented a convolutional neural network architecture for determining whether a given news articles is *biased* (i.e., *hyperpartisan*) or not. In our experiments, we found that a convolutional neural network containing the flatten function and using Google's word2vec embeddings performs best. In the official SemEval 2019 Task 4 test runs, we obtained comparably low precision and accuracy values, while gaining the highest recall rate among all 42 participating teams. For the future, besides evaluating a deeper CNN, we plan to develop two further approaches. The first one will be based on LSTMs. The second model will be a hybrid model, consisting of a CNN layer, which is supposed to learn the text features, appended by an LSTM layer for learning sequence patterns of those features. Furthermore, we plan to work on a non-deep learning approach which assigns controversy scores to news articles and, in this way, determines the bias of the articles.

References

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James R. Glass, and Preslav Nakov. 2018. Predicting Factuality of Reporting and Bias of News Media Sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3528–3539.

Eric Baumer, Elisha Elovic, Ying Qin, Francesca Polletta, and Geri Gay. 2015. Testing and Comparing Computational Approaches for Identifying the Language of Framing in Political News. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT'15, pages 1472–1482.

Edouard Grave, Tomas Mikolov, Armand Joulin, and Piotr Bojanowski. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, EACL'17, pages 427–431.

Felix Hamborg, Karsten Donnay, and Bela Gipp. 2018. Automated identification of media bias in news articles: an interdisciplinary literature review. *International Journal on Digital Libraries*, pages 1–25.

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 2019)*. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP'14, pages 1746–1751.

Craig Macdonald, Iadh Ounis, and Ian Soboroff. 2007. Overview of the TREC 2007 Blog Track. In *Proceedings of The Sixteenth Text REtrieval Conference*, TREC'07.

François Mairesse, Marilyn A. Walker, Matthias R. Mehl, and Roger K. Moore. 2007. Using linguistic cues for the automatic recognition of personality in conversation and text. *J. Artif. Intell. Res.*, 30:457–500.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 Task 4: Sentiment Analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval@NAACL-HLT 2016, pages 1–18.

Iadh Ounis, Craig Macdonald, Maarten de Rijke, Gilad Mishne, and Ian Soboroff. 2006. Overview of the TREC 2006 Blog Track. In *Proceedings of the Fifteenth Text REtrieval Conference*, TREC 2006.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: the detection, classification and structure of arguments in text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 98–107.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2016. Credibility Assessment of Textual Claims on the Web. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, CIKM 2016, pages 2173–2178.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL'18, pages 231–240.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic Models for Analyzing and Detecting Biased Language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL'13, pages 1650–1659.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, SemEval@ACL'17, pages 502–518.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval-2015 Task 10: Sentiment Analysis in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval@NAACL-HLT 2015, pages 451–463.

Dominik Scherer, Andreas C. Müller, and Sven Behnke. 2010. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *Proceedings of the International Conference on Artificial Neural Networks*, ICANN'10, pages 92–101.