Team Kit Kittredge at SemEval-2019 Task 4: LSTM Voting System

Rebekah Cramerus
Department of Linguistics
University of Potsdam
Potsdam, Germany
rebekah.cramerus@fulbrightmail.org

Tatjana Scheffler
Department of Linguistics
University of Potsdam
Potsdam, Germany
tatjana.scheffler@uni-potsdam.de

Abstract

This paper describes the approach of team Kit Kittredge to SemEval 2019 Task 4: Hyperpartisan News Detection. The goal was binary classification of news articles into the categories of "biased" or "unbiased". We had two software submissions: one a simple bag-of-words model, and the second an LSTM (Long Short Term Memory) neural network, which was trained on a subset of the original dataset selected by a voting system of other LSTMs. This method did not prove much more successful than the baseline, however, due to the models' tendency to learn publisher-specific traits instead of general bias.

1 Introduction

With the proliferation of online news agencies after the rise of the Internet, access to information about what is going on in the world has never been more widespread. How that information is presented, however, can have a large influence on what conclusions the reader draws from it. Being able to automatically identify hyperpartisanship (bias or adherence to one party or faction over others) in a news article would help individuals in their news consumption and potentially result in a better-informed population.

As suggested, the challenge approached in this paper is that of hyperpartisan news detection: a binary classification problem (biased or unbiased) with news articles as data. This task can be considered as related to stance detection and in general, sentiment analysis. The challenge was organized as Task 4 for SemEval 2019 (Potthast et al., 2018) (Kiesel et al., 2019). Final submissions were submitted through TIRA, with the test datasets hidden (Potthast et al., 2019).

First in this paper, Section 2 includes an introduction of the provided dataset and a description of preprocessing techniques used for our approach. Section 3 describes the first submitted software, a bag-of-words model. Section 4 continues with our second approach, an LSTM trained on a subset of the original dataset, and a description of how that subset was selected.

Section 5 presents our results on the test set and Section 6 delves into analysis, presenting potential reasons why the models did not perform very well.

2 Data

During the course of the task participants were granted access to different datasets with which to work.

A training dataset of 600,000 articles and a validation dataset of 150,000 articles were both released to participants. Both sets contain 50% unbiased and 50% biased articles, and of the latter, half are left-biased and the other half right-biased (in terms of their placement on the political spectrum). Importantly, these articles were all labeled with the overall bias of their publisher, which was obtained by MediaBiasFactCheck.com and BuzzFeed. The set of publishers whose articles appear in the training set has no overlap with the publishers of the validation set, and neither has any overlap with the publishers whose articles appear in the inaccessible test set.

We consider the labels of these datasets to be noisy: though publishers may have an overall bias, it is likely that most biased news agencies do not publish only biased articles, just as most unbiased news agencies may occasionally publish a biased piece.

Also relevant is a third released dataset referred to in this paper as the *byarticle* dataset. Unlike the other datasets, this one contains articles which were labeled individually through crowdsourcing. It is small, at 645 articles, and unbalanced, at 63% unbiased and 37% hyperpartisan.

2.1 Preprocessing

Certain preprocessing tasks were carried out on the entire dataset pre-training, and also applied to the test set during evaluation.

Some cleaning tasks required segmentation of texts into sentences or words using the Natural Language Toolkit (NLTK) (Bird et al., 2009).

Special characters, double spaces, more than three dots in a row, and any failures in character translation (for example, “gun control” becoming ?gun control?) were replaced or removed. Regular expressions were used for some of these tasks, as well as for removing `img` or `html` tags and URLs. Another list of phrases were summarily removed from each text: those which were likely byproducts of the articles’ retrieval from their websites. These included “Continue Reading Below...”, “Image Source:”, “Opens a New Window”, and so on.

As will be discussed, a recurring problem encountered by our models was the tendency to learn publisher-specific traits and not hyperpartisanship itself. To combat this we included methods to remove potential publisher-specific text, especially names and emails of the authors of the articles, in our preprocessing step.

3 Software 1: Bag-Of-Words

Our first approach was a bag-of-words baseline for initial exploration of the dataset and comparison with the more complex second approach.

3.1 Tokenization / Lemmatization

Texts were tokenized into words, and the words reduced to their lemmas, using spaCy (Honnibal and Johnson, 2015).

3.2 Vectorization

First we created a vocabulary of the most common 4000 words in the overall corpus (all datasets), excluding stopwords from a list compiled by scikit-learn (Pedregosa et al., 2011).

We also excluded from the vocabulary words from an exceptions list, in an attempt to reduce the problem of overfitting to the article publishers. This exceptions list was formed by counting all words in the training and validation datasets, and gathering those words which appeared five times more often (relative to the size of the corpus) in one set than in the other. Some of these terms were location-specific (*abq*, *lobos*, *nmsu* — likely a publisher based in New Mexico) and others hinted at coverage of a certain topic (*samsung*, *boeing*, *verizon* — possibly a publisher which wrote often about the stock market). The intent behind this was to help avoid the model picking up, for example, that the presence of terms surrounding New Mexico automatically meant a certain label.

With a final vocabulary, each text was then converted to a vector of length 4000, wherein each dimension is the count of the corresponding vocabulary word in that text.

3.3 Model

Using scikit-learn, the training and validation datasets were vectorized according to the previously described specifications and then fit to a logistic regression model. This was then submitted to TIRA.

4 Software 2: LSTM

Long Short Term Memory networks (or LSTMs) are a form of Recurrent Neural Networks (or RNNs) which is capable of learning long-term dependencies. Given the nature of the problem and the data — a text being a sequence of words, the relationship between them as important as the words themselves — we chose to develop a model with this architecture.

4.1 Word Embeddings

To transform the article texts into vectors able to be processed by the neural network, we chose to train our own skip-gram word embeddings on the total given corpus (training, validation and byarticle datasets). Embeddings of 50 dimensions (chosen mostly arbitrarily, but in part due to computational limits) were trained using the Python package gensim, for 10 epochs, including words in the vocabulary which appeared in the corpus over five times (Řehůřek and Sojka, 2010). The total vocabulary size was 457,197 words.

4.2 Vectorization

Texts were first transformed into arrays of the shape (100, 50), wherein 100 was the cutoff or maximum text length and 50 was the dimensionality of the word embeddings. Texts shorter than 100 words were padded with zero-vectors to keep the shape consistent to feed into the network.

Model	Test Dataset	Accuracy	Precision	Recall	F1
Bag-of-Words	byarticle	57.8	54.7	90.8	68.3
LSTM	byarticle	58.3	55.8	79.2	65.5
Bag-of-Words	bypublisher	61.2	57.8	83.4	68.2
LSTM	bypublisher	65.2	64.7	67.1	65.9

Table 1: Results on the test datasets.

4.3 Architecture

The model consists of a single LSTM layer with 50 units, followed by a dropout wrapper with a keep probability of 0.75 to help prevent overfitting. Next is a standard feedforward neural network output layer. AdamOptimizer was used with a 0.001 learning rate as well as softmax cross-entropy loss for optimization.

All LSTM models were trained using Tensorflow for approximately 2 epochs (Abadi et al., 2015).

4.4 Voting System

Knowing that the biggest obstacle faced so far was the tendency of models trained on the datasets to overfit to the publishers and not bias itself, we chose to pare down the dataset for the final submission. In theory the ideal dataset, in which there is no noise from the publisher-based labels, is contained within the original dataset. To find that subset — or at least to get closer — we implemented a voting system.

Three LSTMs of the previously described architecture were trained: one each on the training, validation and byarticle datasets. We then collected predictions from each LSTM, on each article in each dataset. The articles which all three LSTM models correctly labeled were pulled into a new dataset labeled *agree*. This dataset, in total size 162,046 articles with 37% biased and 63% unbiased labels, was what we trained our final model on.

4.5 Retrained LSTM

Once the new datasets were compiled from the voting system based on the originals, a new LSTM with the same architecture was trained on the combined data. This model was submitted to TIRA as our second software.

5 Results

Both approaches were scored on the hidden test datasets using TIRA. One of the two test datasets was labeled individually by article, referred to in this paper as the *byarticle-test* dataset, including 638 articles with no publisher overlap with any of the given corpora. The other, like the training and validation sets, was labeled overall by publisher, here referred to as the *bypublisher-test* dataset, with a total of 4000 articles, also including no publisher overlap with other datasets.

Results can be seen in Table 1. The LSTM tended to outperform the Bag-of-Words model in accuracy, but had lower f1 scores. All results showed higher recall than precision — and except for the case of the LSTM with the bypublisher-test set, markedly higher. By accuracy, the best result was the LSTM on the bypublisher-test set.

6 Discussion

In the published leaderboard, most teams had higher scores on the test dataset which was labeled by article rather than the one labeled by publisher; overall, the highest accuracy was over 10% higher on the byarticle-test set than on the bypublisher-test one. Our approaches, on the other hand, both performed better on the bypublisher-test dataset. This could be in part because we did not spend too much time optimizing over the small byarticle dataset which was released to us — trying additional techniques to maximize performance over this set could be a task for future work.

Why our results are better on the bypublisher-test sets is an interesting question. Our efforts in both approaches were focused on enabling the models into generalizing about bias, instead of on recognizing only which articles belong to which publishers. Better performance on the bypublisher-test run than on the byarticle-test run suggests that our efforts may have paid off, but in the sense that we are better able to identify biased publishers instead of biased articles. That is, the question that the first models were answering was, “Does this article belong to X set of publishers, or Y set of publishers?” We attempted to instead answer the question, “Is this article biased?” But higher accuracy on the bypublisher-test dataset indicates that we might instead answer the question, “Does this article belong to a biased *publisher*?”

6.1 Precision/Recall

Across all models recall was consistently higher than precision. Our approaches therefore were correctly picking out hyperpartisan articles, but also misclassifying unbiased articles as biased. There are a variety of reasons why this could happen: quotes by partisan speakers could affect a rating of an unbiased article which discusses it, or certain topics could be more often reported in a partisan manner so that unbiased articles around them are rare and misclassified. A closer examination of the test dataset results would be needed for a more concrete discussion.

6.2 Software 1 Exceptions List

The exceptions list was created with the intent of removing words from the vocabulary which were extremely lopsided in their use between publishers (as in, some publishers, or just one in particular, were much more likely to use the term than others). While initial results looked promising, it is possible that the method was not robust enough, and some unigram indicators of a certain publisher still ended up in the final model.

6.3 Software 2 Dataset: Voting System

The idea behind the voting system was to pare down the original dataset, reducing noise and therefore focusing on the data points where bias was most salient — and could as such be picked up by models trained on different publishers. Theoretically these articles would all have characteristics common to biased articles of all publishers. When put together, then, the hope was that a model trained on this subset of the dataset would learn those common characteristics and not just the publisher-specific ones.

There are many things that could have gone wrong with this system, however. Our cutoff for the news article length may have been too short, for example. Secondly, Since the three models used for voting did not have very high accuracy on each other’s datasets in the first place, the level of noise may not have been reduced at all. Furthermore, the original training dataset was four times as large as the validation dataset, and the byarticle dataset was far smaller than either. Their subsets after the voting system was applied were equally unbalanced. When combined and used for training the final LSTM, it could have been unbalanced enough that the model learned mostly from the data from the original dataset, and the features from those publishers.

7 Conclusion

In approaching Task 4 (Hyperpartisan News Detection) in SemEval 2019, we developed two models for submission: a Bag-of-Words logistic regression model and an LSTM neural network trained on a subset of the original training and validation sets. While neither model reached high accuracy rates on the test datasets, their methods still provoke some discussion on how to better avoid fitting to the publishers and not bias itself.

8 Namesake

Margaret Mildred “Kit” Kittredge is a character from the American Girl doll and book series. She was born in Ohio in the 1920s and wanted to become a reporter when she grew up. In her room in the attic, she would write her news articles on a typewriter to share with her family.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Steven Bird, Edward Loper, and Ewan Klein. 2009. Natural language processing with python. O’Reilly Media Inc.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

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.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

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

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.