CS230

Identifying Political Spectrum in News Articles - deepnews.ai -

Markus Zechner
mzechner@stanford.edu

Halldora Gudmundsdottir
halldora@stanford.edu

Abstract

Digital journalism has increased the accessibility to news articles drastically. The sheer amount of information is a critical challenge for readers. This project aims to classify news articles based on their political spectrum from liberal to conservative, to improve recommendations for readers. We showed that a LSTM can lead to high accuracies but is limited due to long training times. A one dimensional convolutional network architecture with max pooling layers and dropout was able to achieve the same accuracy and decrease training time by a factor of 4. Time limitation only allowed us to train the models on a limited data set but two experiments on the full data set showed that a accuracy of 90% can be achieved with the convolutional network. An error analysis were an experienced journalist labeled a small set of articles showed that the labeling based on publishers could lead to 38% of mislabeling.

1 Introduction

While digital journalism has increased the accessibility to news articles it also provides readers with constant flow of information, most of which is of little interest to readers. The main goal of this project is to automatically classify news articles based on their political spectrum which ranges from liberal to conservative. But what are the direct implications of ranking news article based on the political spectrum? By automatically identifying the political bias of an article, the recommendation engine for users could be improved. People reading many liberal articles are likely to prefer liberal articles in the future as well. Furthermore, it could be used to offer users a variety of news articles, from liberal to conservative, in order to provide the users with different angles for a specific story.

The input to our models are text bodies of articles from various news publishers. The articles are cleaned and processed and the models trained on the raw text after applying pre-trained word embeddings (GloVe). Based on this input, articles are classified into three categories: liberal (left), conservative (right) and neutral (center). We compare the performance of several models: (1) a simple softmax regression baseline, (2) a standard fully connected neural network, (3) a LSTM model, (4) a CONV1D model and (5) a combination of CONV1D and LSTM. The model codes can be found on `https://github.com/MarkusZechner/CS230-Project`.

2 Related work

The main goal of this project was text classification and thus our initial efforts were spent researching RNNs, given that text is naturally sequential. The latest improvements in RNNS served as guidelines with special focus on LSTM models for text classification and sentiment analysis [4, 9]. In literature, the state-of-the-art NLP approaches often switches between CNNs and RNNs. CNNs are good at extracting position-invariant features like key phrases while RNNs are more appropriate for context dependencies. For our project, computational constraints limited training of RNNs and recent studies

CS230: Deep Learning, Spring 2018, Stanford University, CA.

show that CNNs often provide equivalent information for text classification. Which architecture performs better depends mostly on the importance of understanding the whole text sequence [1, 2, 8].

Other areas of research are hybrid models of CNNs and RNNs as well as Attention models. CNN-RNN models are used to address the limitations of the individual models, where the CNN portion reduces the number of parameters and the RNN portion captures contextual information [3, 6]. Attention models have been used to assess which parts of the text body are critical for the classification task, since all parts might not be equally relevant [7, 10]

3 Dataset and Features

The dataset of news articles was provided by deepnews.ai. The dataset is a collection of news articles that originate from different publishers ranging from very liberal to very conservative. As human labeling is expensive, the political spectrum of publishers was analyzed using the website `www.allsides.com`. Therefore, articles are labeled based on their publisher and not individually read and classified. For example, all articles obtained form the New York Times are labeled as liberal because of the general political bias of the publisher. Table 2 in Appendix summarizes the different publishers, the number of articles and their corresponding political spectrum used for our study. Because of the constraints of computational capabilities and working with a balanced dataset, we focused on using the smaller dataset (Dataset 1) for developing and tuning our models. We then used the larger dataset (Dataset 2) to test our models. Since our data are in the order of ten-thousands we used a train/dev/test split of 70/15/15%.

The data came in various formats and required different levels of preprocessing. We developed a pipeline to extract the various articles (.txt, .csv files) and clean repeating sentences, advertisements and other insubstantial material. Moreover, we applied the following cleaning steps: (1) separate all special characters from words (e.g. commas, colons), (2) remove all special characters, (3) convert letters to lower case, (4) remove extra white spaces, (5) split each sentence to words. For word representations we used pre-trained 50 dimensional GloVe embeddings [5]. These vectors were then used as inputs to our models.

4 Methods

For our baseline we used a simple softmax regression. An average (d_{avg}) was found across all word vectors in an article to get a single 50 dimensional vector representing that article, $d_{avg} = \frac{1}{N}\sum_{i=1}^{N} w_i$, where N is total number of words and w_i the GloVe vector for word i. This vector was then fed into an output layer with a softmax activation function to obtain predictions.

All models introduced in this section were trained using categorical cross-entropy loss with Adam optimizer and evaluated with an accuracy metric (% of correctly classified articles). The reasons behind choices, advantages and limitations of models are described in Chapter 5. Illustrations of model architectures can be found in Figure 5 in Appendix.

4.1 Fully Connected (FC) Neural Network

For the fully connected neural network we used 3 hidden layers, each with 500 units and ReLu activation, and an output layer with a sigmoid activation. Similar to the baseline, the input was an averaged vector of all GloVe word embeddings for a particular article. The hyperparameters we tuned for this model were learning rate, batch size, #hidden units, #layers and dropout rate. For the best model we used a learning rate of 0.0001, batch size of 128 and dropout rate of 0.5.

4.2 LSTM Recurrent Neural Network

The LSTM model was a 2 layer model where each LSTM cell had a hidden size of 128. The LSTM model was trained to predict the political bias of a sequence of 1000 words from articles. The output from the LSTM model was fed into a fully connected layer with a sigmoid activation to obtain predictions. To prevent overfitting, we used dropout for regularization. Because LSTM took a long time to train, rigorous hyperparameter tuning was not possible, but we manually tested several parameters such as #LSTM layers (1 and 2), size of LSTM cells, article length (# of words) and

dropout rate to obtain a good model. For the best model we used a learning rate of 0.001, batch size of and dropout rate of 0.5.

4.3 CONV1D Convolutional Neural Network

Our CONV1d model consisted of 4 one-dimensional convolutional layers, each with 128 filters (size=5, stride=1). To reduce the dimension of the parameter space, a max-pooling layer followed each convolutional layer (size=5, stride=3). The output was then flattened and fed into a fully connected layer of 128 hidden units with ReLu activation, followed by an output layer with softmax activation for final predictions. Dropout was used to address overfitting. The hyperparameters we trained for this model were learning rate, batch size, article length (# of words), # filters ad dropout rate. For the best model we used a learning rate of 0.001, batch size of 128 and dropout rate of 0.5.

4.4 CONV1D-LSTM Combination

In order to combat limitations of LSTM and CONV1D we also considered a combination of the two models. The CONV1D-LSTM contained 2 one-dimensional convolution layers followed by max-pooling layers. The convolutional layers had 128 filters of size 10 and the max-pooling layers were of size 2 with stride of 2. The output from the convolutional part was fed into 1 layer of LSTM cells (of size 128) followed by an output layer with softmax activation for final predictions. Dropout was used to address overfitting. The hyperparameters we trained for this model were learning rate, batch size, article length (# of words), # filters ad dropout rate. For the best model we used a learning rate of 0.001, batch size of X and dropout rate of 0.5.

5 Results and Discussion

5.1 Hyperparameter Tuning

An essential part in deep learning is it to find a set of hyperparameters that maximize the chosen metric (in our case accuracy). The sheer number of parameters and their wide range span a vast range of possible combinations to cover in order to find an optimum set. Their interactions (e.g. learning rate and batch size) makes this evaluation even more involved. The class taught us to randomly sample the parameter space to identify influential parameters and to find a set of parameters that result in a high evaluation metric. After an initial random search, the parameter space is narrowed down in order to refine the search space. The main limitation of this approach is the number of experiments needed to cover the parameter space and because our models had very long running times it was not deemed feasible. After discussions with the teaching team we decided to follow a staged approach where we tested each hyperparameter individually and fixed it when moving to the next one. The main constraint of this approach is that interactions between hyperparameters are not considered. For example learning rate and batch size influence each other and therefore optimal combinations might be missed since learning rate is fixed in the first stage. For our models we generally tuned the hyperparameters in the following sequence: learning rate, batch size, article length, architecture (#hidden units, #layers, #filters) and dropout rate. The best hyperparameters for each model are reported when describing the model architectures in Chapter 4. An example of the hyperparameter tuning process for our CONV1D model is illustrated in Figure 1. Overall the most sensitive hyperparameters were the learning rate and article length.

5.2 Performance of Models

The performance of the best models from our hyperparameter tuning process are reported in Table 1. The baseline was able to predict with 58.5% accuracy, which is little better than random guessing. By applying a standard fully connected (FC) network we were able to increase the accuracy to 67.7% but the model was not able to capture the full complexity of the articles. Moreover, by averaging all word vectors for an article, much information is lost that is important for accurate classification.

Because of the sequential text structure of news articles, our first attempt to improve the accuracy of the classification task was to implement a LSTM recurrent network. LSTM work well for inputs with contextual dependencies but take a very long time to train. Therefore, we were not able to hyper-tune the LSTM structure but still achieved an accuracy of 84.4%. In order to speed up training, we next

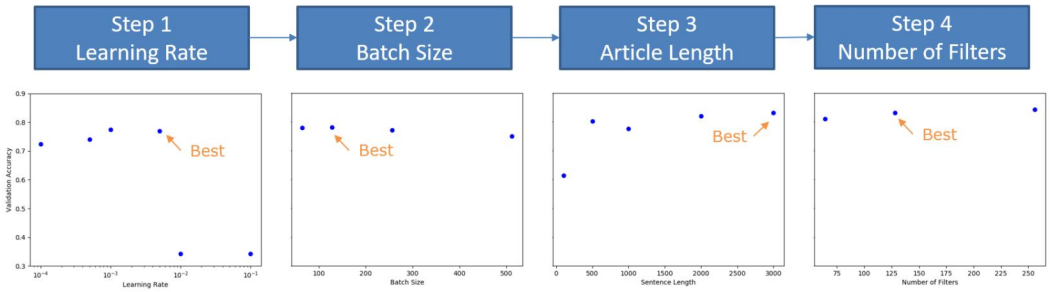

Figure 1: Hyperparameter tuning for the CONV1D model.

considered a convolutional network (CONV1D). The CONV1D model took a short time to train (<1.5 hrs), allowing us to hyperparameter tune it, and gave surprisingly high accuracy of 84.3%. The fact that the CONV1D and the LSTM have similar accuracy can be accounted to the fact that the LSTM only only considers context in the vicinity but cannot establish contest over the course of thousands of words.

Lastly, we explored if we could combine the best from both models, by reducing the input of the LSTM through a CONV1D architecture. The combined hypertuned architecture showed a similar performance of 83% for the same reasons mentioned above.

As can be seen in Table 1 we experienced overfitting in all our models, even when applying dropout for regularization. We identified the size of the training sample to be the major contributing factor. Due to computational constraints, the models were tuned using a smaller dataset (Dataset 1, see Table 2). However, when we applied our models on the larger dataset (Dataset 2) we saw great reduction of overfitting and increased accuracy, yielding CONV1D the highest performing model with 90.2% accuracy. The training curves and regularization effects for CONV1D are illustrated in Figure 2. Corresponding loss curves can be found in Figure 4 in Appendix.

Table 1: The performance of models after hyperparameter tuning.

Model	Training set		Validation set		Test set		Training	Dataset	Tuning
	Loss	Acc.	Loss	Acc.	Loss	Acc.	Time		
Baseline	0.88	58.8%	0.89	58.5%	0.88	58.0%	1.6 min	1	Manually
FC	0.72	68.7%	0.73	67.6%	0.73	67.7%	10 min	1	Sequentially
LSTM	0.13	95.2%	0.61	84.4%	0.63	84.1%	15.8 hrs	1	Manually
CONV1D	0.16	93.8%	0.59	84.3%	0.60	83.8%	1.4 hrs	1	Sequentially
	0.23	**91.3%**	**0.29**	**90.2%**	**0.29**	**90.3%**	5.5 hrs	2	Not tuned
CONV1D	0.29	88.6%	0.45	83.0%	0.46	82.1%	5.6 hrs	1	Sequentially
+ LSTM	0.18	93.1%	0.29	89.4%	0.23	89.9%	18.9 hrs	2	Not tuned

5.3 Error Analysis

To get more insights into the misclassification, we decided to perform an error analysis. Since we are not experts in journalism we asked an experienced journalist to classify 24 mislabeled articles (Table 3 in Appendix). The publisher and the label of the article was not shown to the journalist until the assessment was finished. The journalist labeled 62 % of the misclassified articles 'correctly'. We have to be careful with the term 'correctly' because it assumes that our labeling (based on publishers) is correct. We can also look at this percentage of 62% differently. If we assume that the journalist's judgment of the articles revealed the real ground truth label of an article, 38% of the articles got mislabeled by choosing the publisher as the label. We think that moving forward with this classification, human labeled articles are essential. In general, the publisher is a good proxy for the political bias label but every newspaper features both liberal and conservative articles. This could lead to mislabeling, when only considering the publisher as the label.

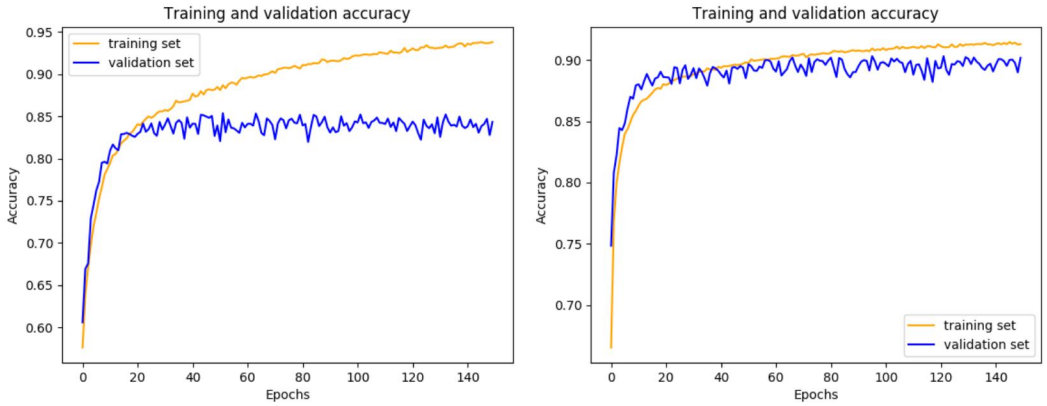

Figure 2: Accuracy curves for the CONV1D model, using **Dataset 1** (left) and **Dataset 2** (right).

A discussion with the journalist revealed that the distinction between conservative and liberal article was fairly easy for him, whereas the classification between center and conservative or liberal articles was much more involved. This can be seen in the confusion matrix for the journalist in Figure 3. The journalist approaches the classification task by finding political messages that can be linked to a political party. He mentions that this task involves tracking phrases throughout the entire article. In some cases, certain strong statements lead to an immediate conclusion about the political bias. The fact that phrases across the entire article are used by a human to classify an article are an indication why the LSTM architecture did not perform much better than the convolutional network. We think that the length of articles (thousands of words) are too long for current LSTMs to bring critical words and phrases in contact with each other.

Looking at the confusion matrix for our best model (CONV1D) in Figure 3, most of the misclassified examples are between conservative and liberals. This is very counterintuitive since a human performs much better in that task. This likely originates from the same limitation of LSTMs mention in the previous section. A liberal article as well as a conservative article could talk about the same topic, but the political bias can only be assessed by bringing the entire article into context. For example, many articles are talking about president Trump but only the context of the article lets the reader conclude about the political opinion.

The journalist also indicated that center articles can be subdivided into two very distinct classes. A ‘center’ labeled article could be neutral because it only reports facts. In contrast to this an article could also be ’center’ because it views a topic from different political angles in a very balanced way. It could be challenging for an algorithm to classify value-neutral as well as political balanced articles with the same label. In this context, a large number of quotes could be used as an indication for a more ’center-factual’ articles.

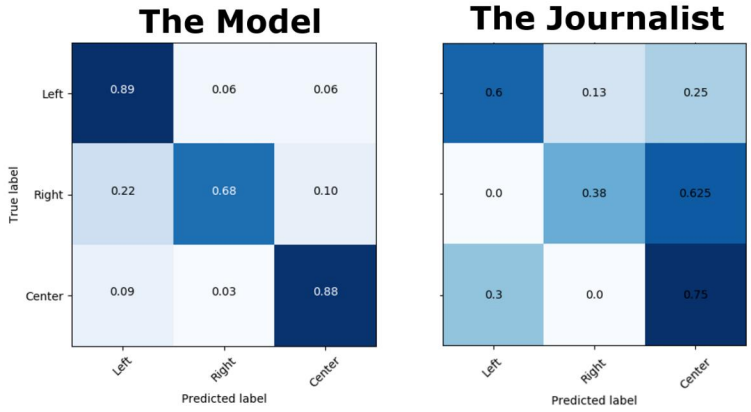

Figure 3: Normalized confusion matrix for the best model (left) and the journalist (right).

6 Conclusion and Future Work

The highest performing algorithm in our study was the CONV1D convolutional network with an accuracy of 90.2%. Computational constraints limited tuning of our LSTM models, but initial investigation suggested that articles were too long for the LSTM to identify dependencies of critical words and phrases. The error analysis revealed that our model had difficulties classifying between conservative and liberal articles but center ones were easier. This highlighted the limitations of our initial labeling of the dataset.

For future work, we would like to perform a more detailed error analysis. The error analysis was done on a limited sample size (24 articles) but more examples (>100), classified by several journalists, could substantiate the conclusions drawn. This extended error analysis will allow us to quantify the mislabeling procedure on a publisher basis (see section 5.3). For example, classifying a New York Times article as conservative would be incorrect based on our labeling, but if several journalists classified the same article as conservative, the classification of the article would be correct. Here, an understanding of the origin of misclassification could be developed: did our model just get it wrong or was the article wrongly labeled initially? Depending on this, we could then use the 'misclassification' as a hint for the new label. Labeling was by far one the most critical limitations in this work and obtaining journalist labeled articles would greatly enhance the model.

We also recommend gaining more insights into how an article is classified, or in other words - which words or phrases are critical in determining the political direction of an article. An attention model would be a good approach to answer those questions. Our literature survey for LSTM attention models showed that this topic is new but a good starting point could be the hierarchical attention model proposed by Yang et. al [7]. These insights could also be compared with important phrases identified by journalists.

7 Acknowledgments

The authors of this project would like to thank Zahra Koochak and Lucio Dery for their valuable comments and discussions. We also would like to thank the entire CS230 team for the teaching and Amazon Web Services for providing computational resources. Special thanks go to Ferderic Filloux and Mathieu Delcluze for providing us the data, giving us insight into the world of journalism and for their valuable discussions.

8 Contributions

All team members contributed equally to this project.

Appendix

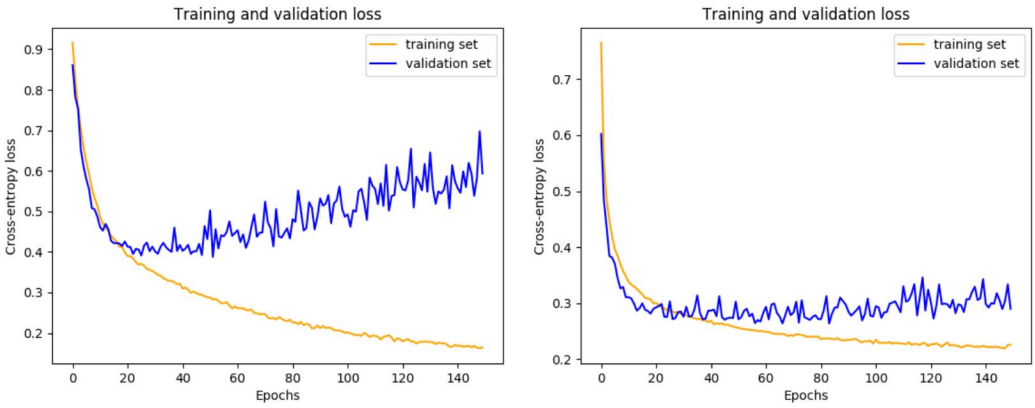

Figure 4: Loss curves for the CONV1D model, using **Dataset 1** (left) and **Dataset 2** (right).

Table 2: News article collection with media bias ratings.

Publisher	Political Bias	Dataset 1	Dataset 2
New York Times	Liberal	3,340	7,799
Guardian	Liberal	3,340	25,914
Washington Post	Liberal	3,340	11,002
Atlantic	Liberal	3,340	7,113
Breitbart	Conservative	3,340	23,375
Fox News	Conservative	3,340	4,324
National Review	Conservative	3,340	6,144
New York Post	Conservative	3,340	17,466
Reuters	Neutral	3,340	10,710
Quartz	Neutral	3,340	17,132
Financial Times	Neutral	3,340	17,134
Business Insider	Neutral	3,340	6,332
Total		40,080	154,445

Table 3: Summary of the error analysis with the journalist.

Publisher	True Label	Predicted Label	Journalist Label	Journalist Comment
Atlantic	Liberal	Conservative	Center	factual article
Breitbart	Conservative	Liberal	Center	factual article
Washington Post	Liberal	Conservative	Center	factual article
National Review	Conservative	Liberal	Conservative	
Guardian	Liberal	Conservative	Conservative	
New York Times	Liberal	Conservative	Liberal	
New York Times	Liberal	Conservative	Liberal	this is a book review
Fox News	Conservative	Liberal	Center	factual article
New York Post	Conservative	Liberal	Center	factual article
New York Post	Conservative	Liberal	Center	factual article
National Review	Conservative	Liberal	Conservative	
Washington Post	Liberal	Conservative	Liberal	
Fox News	Conservative	Liberal	Center	factual article
Atlantic	Liberal	Conservative	Liberal	
Breitbart	Conservative	Liberal	Conservative	
Guardian	Liberal	Conservative	Liberal	
Reuters	Center	Liberal	Liberal	factual article
Reuters	Center	Conservative	Center	factual article
Business Insider	Center	Conservative	Center	factual article
Business Insider	Center	Liberal	Center	factual article
Finanical Times	Center	Liberal	Center	factual article
Finanical Times	Center	Conservative	Center	factual article
Quartz	Center	Liberal	Liberal	just slightly liberal
Quartz	Center	Conservative	Center	factual article

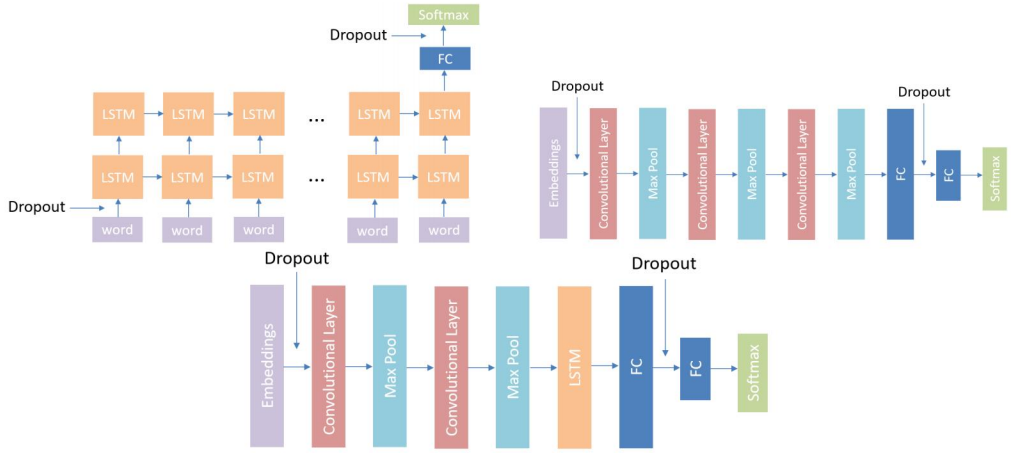

Figure 5: Model architectures: LSTM (top left), CONV1D (top right), CONV1D-LSTM (bottom center).

References

[1] R. Johnson and T. Zhang. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Advances in Neural Information Processing Systems*, pages 919–927, 2015.

[2] Y. Kim. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014.

[3] S. Lai, L. Xu, K. Liu, and J. Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015.

[4] T. Nasukawa and J. Yi. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2Nd International Conference on Knowledge Capture, New York*, 2003.

[5] J. Pennington, R Socher, and C.D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[6] Y. Shen, H. Yun, Z.C. Lipton, Y. Kronrod, and A. Anandkumar. Deep active learning for name entity. In *Published as a conference paper at ICLR 2018*, 2018.

[7] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North America Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

[8] W. Yin, K. Kann, M. Yu, and H. Schutze. Comparative study of cnn and rnn for natural language processing. *CoRR*, 2017.

[9] Y. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing. *CoRR*, 2017.

[10] H. Yu and C. Yue. News article summarization with attention-based deep recurrent neural networks. 2017.