CS230

Topic Analysis with Deep Learning on Large Documents

Nathan Zhao
Department of Applied Physics
Stanford University
nzz2102@stanford.edu

Jiahui Wang
Department of Applied Physics
Stanford University
jiahuiw@stanford.edu

Xianling Long
Department of Management Science and Engineering
Stanford University
xianling@stanford.edu

Abstract

We test a variety of deep-learning techniques in convolutional neural nets and attention-based recurrent neural nets to analyze topic structure and content in large corpuses of digital media from two large reputable news organization: the *Guardian* and the *Economist*. We achieve what appears to be state of the art results on topic recognition.

1 Introduction

Classification of text has been a widely explored and important area of deep learning research. Being able to automatically map the content of text to certain labels have significant applications ranging from spam-filtering to, more recently, fake news detection. In addition, as many traditional forms of readable media, such as news have become increasingly digital, the abundance of textual data describing the daily events of the world have become readily available. One problem is that the proportion of well-labelled data to unlabelled data is very lopsided. Using deep-learning, we hope to train a successful algorithm to classify news articles based on selected topics. The input to our algorithm is an article in *Guardian* or *Economist*. We then use a multi-pooled CNN and a bidirectional RNN to output a predicted topic of the article.

2 Dataset and Features

We test our algorithms on a corpus of 10,000 topic-labelled articles classified into 50 classes from *Guardian*, a prominent news service based in the United Kingdom, and a corpus of 95,000 topic-labelled articles classified into 38 classes from Economist, a practitioner in the social science discipline of economics. The *Guardian* also has 150,000 unlabelled articles which is the Guardian Deployment dataset.

Table 1: News Corpus Dataset Statistical Summary

Dataset	# Documents	Unique Tokens	Avg. Word Count	Labels
Guardian Train/Test	10000	102,000	675	27
Guardian Deployment	150000	–	841	–unlabelled–
Economist	95000	395,000	740	21

For the *Guardian*, analysis of the labels shows a very non-uniform distribution of the examples between the 50 topic labels. In particular, 24 of the classes have 50 articles or less, so we group all articles in these 24 classes into one collective label called ‘OTHER’, which narrows our label-space to a 27-category classification problem. A sample of the remaining labels are: business, world news, games, arts, environment, technology, society, etc. For Economist, we also perform the same label condensation technique, which brings the number of articles to 21. We also note that part of the condensation requires a manual inspection of the labels as some topics are clearly highly overlapped such as "Brief" and "News Brief".

To show that the text corpora are non-trivially distributed, we use t-distributed stochastic neighbor embedding (t-SNE) to visualize the corpus using the term-frequency inverse document frequency (tf-idf) matrices for the different datasets:

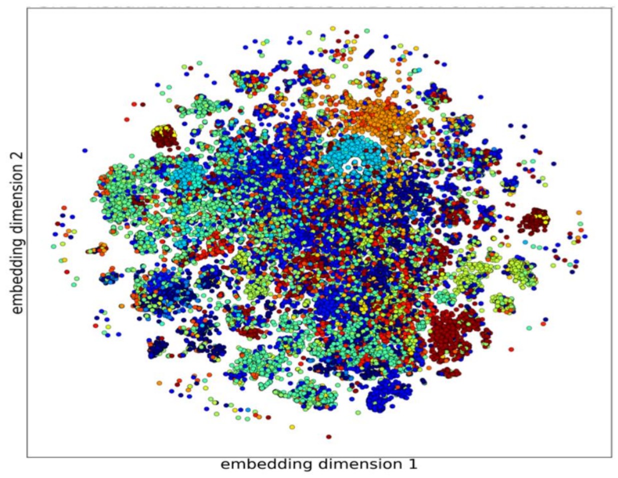

Figure 1: Topic Distribution for Economist

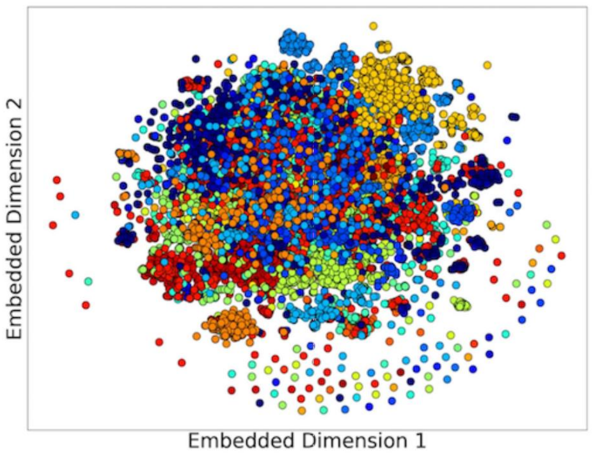

Figure 2: Topic Distribution for Guardian

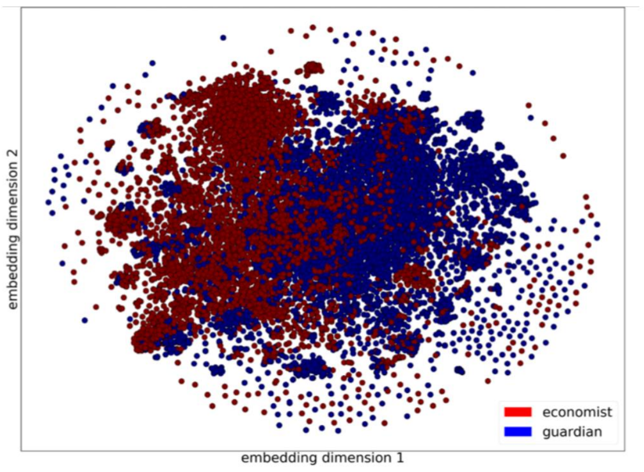

Figure 3: Guardian vs. Economist Articles

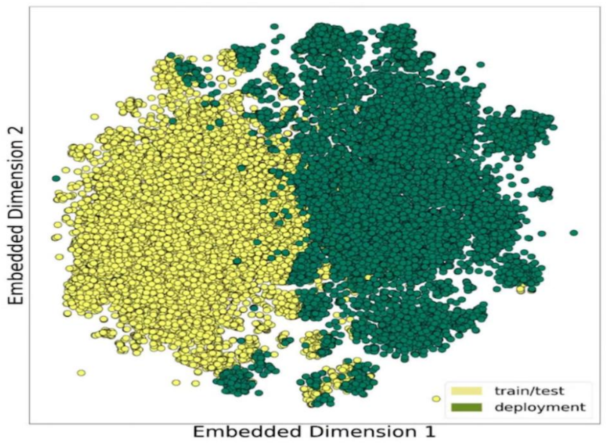

Figure 4: Guardian vs. Deployment Articles

The preprocessing of the data involves converting the data from .csv format into a list of strings, with each strings representing one document. For each document, we also remove stop-words, lemmatize, and normalize by converting to lower-case. We observe that usual considerations which might trivialize NLP tasks such as headers, footers, and metadata are not usually a problem as the datasets have already separated the text body and the metadata.

From there, we tokenize the entire corpus, which generates a bijecive map of all the words to a positive integer. From there, we convert each document in the corpus to a sequence where each position represents a word and is occupied by a positive integer corresponding to the word that was originally in that location of the document.

3 Methods

We try three different algorithms on our dataset. To benchmark the performance of deep learning algorithms, we use Naive Bayes, the traditional workhorse algorithm for NLP tasks. We try two types of neural nets: convolutional neural nets and recurrent neural nets, specifically long short-term memory networks (LSTM).

3.1 Benchmark Naive Bayes

Naive Bayes is a simple generative model which estimates the conditional probability that we are usually interested in $p(y|x)$ by modeling the joint probability $p(y,x)$ as a product (assuming essentially that the appearance of each word is independent of the others, which is a bad assumption for NLP):

$$p(y,x) = p(x_1|y)p(x_2|y)...p(x_n|y)p(y) \tag{1}$$

Surprisingly, Naive Bayes algorithms do not perform badly in many tasks as we will demonstrate later in our results. We also note that as Naive Bayes only requires word frequencies and document frequencies per label in training, the input is just the tf-idf scores.

3.2 Convolutional Neural Net

The intuition behind using a convolutional neural net is based partly on the fact that the documents in the digital media corpuses are relatively long. While CNNs have been shown to work on text classification tasks, we believe they are particularly applicable here as long documents tend to have a hiearchical structure that goes beyond individual words and sentences which we believe convolutions and pooling can help quantify. With that said, the convolutional neural networks that we deploy generally contain four key elements:

1. Word Embedding layer: For our word embedding, we try two techniques. The first is to actually train the word embedding like the other layers. The second is to use a pretrained word embedding such as GloVe or Word2Vec. Work in (Yoon, 2014)[1] has shown that using pretrained word embeddings often leads to superior performance in CNNs. However, we observe the opposite (and will explain down below)
2. Convolutional layers: As shown in Figure 5, we use three one-dimensional convolutional layers with filters of different sizes (sizes = [3,4,5]) and 100 filters for each layer. Then we use max-pooling to extract the most important features.
3. Dense layers: Our current optimal CNN has no hidden dense layers except the softmax layer to predict the output. Our basic parameter scan shows that adding hidden dense layers tend to decrease overall accuracy on the dev and test sets.
4. Softmax classification

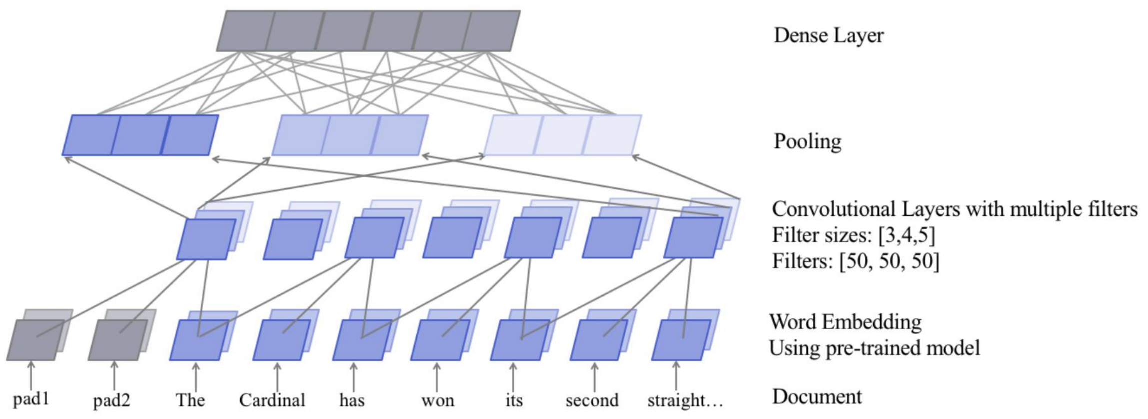

Figure 5: Architecture of our CNN algorithms. To gain information at different levels of coarseness, we preferentially pool multiple convolutions into a single layer rather than doing a sequential set of convolutions with different filter sizes

3.3 LSTM

We deploy bidirectional LSTM with and without attention model. The structure of the LSTM is holistically the same as the CNN model but instead of doing convolutions, we feed the document as a sequence of words into a series of LSTM units.

3.3.1 Attention

The LSTM with attention is a significant modification on the original LSTM, deploying the usual encoder-decoder network with the attention module inserted in between. However, instead of an output to a sequence as in machine translation, we direct the output into a final softmax layer which produces a prediction of the class that the document should be in.

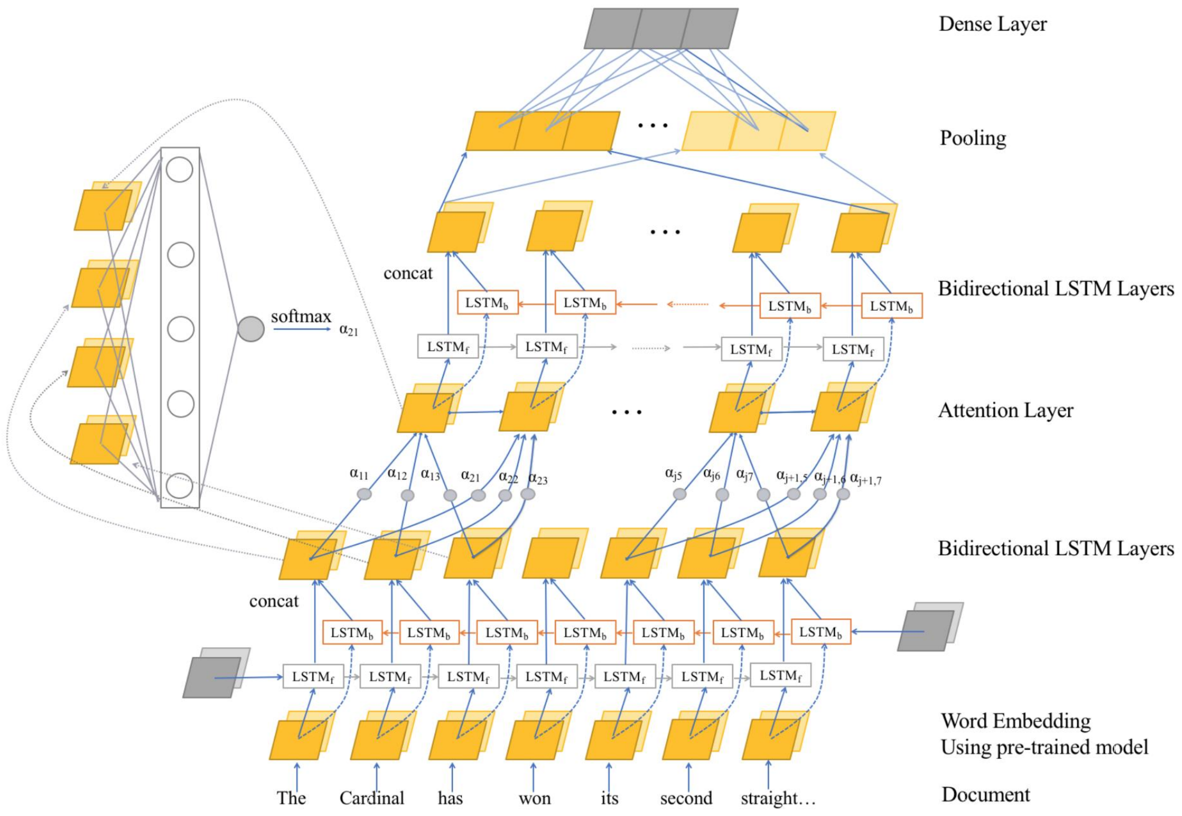
Dense Layer
Pooling
Bidirectional LSTM Layers
concat
softmax
α_{21}
LSTM$_b$
LSTM$_f$
Attention Layer
α_{11} α_{12} α_{13} α_{21} α_{22} α_{23} α_{j5} α_{j6} α_{j7} $\alpha_{j+1,5}$ $\alpha_{j+1,6}$ $\alpha_{j+1,7}$
Bidirectional LSTM Layers
concat
Word Embedding
Using pre-trained model
The Cardinal has won its second straight…
Document

Figure 6: Architecture of our RNN algorithm.

4 Experiments/Results/Discussion

Naïve Bayes already shows pretty good accuracy on the test set, which intuitively suggests that on a per-topic basis, certain words must generally appear together for an article about any given topic. Using a CNN clearly performs better by almost 30 versus our NB benchmark and is quite good considering that the classification task is quite difficult (27 labels). Currently, we also have two interesting observations about the CNN performance. The CNN performs better without any dense layers in it where adding a small dense layer of 64 nodes decreases the test accuracy by about 3% to 84%. More dramatically, the CNN performs better in training its own word embedding rather than using a pre-trained word embedding such as GloVe (71% test accuracy), which is contrary to our expectations. So in two respects, the CNN architecture which does best for the Guardian dataset is different from what your usual literature review would suggest. Presently, we do not have a good explanation for it, but we believe the fact that the relatively long lengths of the articles may be a factor. LSTMs actually do not perform at the same level as our CNNs. Adding attention seems to improve it for one but not the other

Table 2: Summary of Algorithm Performance (Accuracy %)

Algorithm	Guardian	Economist
Naive Bayes	68	55
RNN	82	72
CNN	87	85

4.1 Related Work

Now we compare our results with some similar topic classification results. In (Yoon, 2014) [1], performances in the range of 75% to 95% are achieved on classification tasks, but with many fewer labels (two to six) than our task demands. Additionally, the Stanford NLP group has achieved 88% accuracy on a common dataset called the 20 Newsgroups dataset (20000 articles with 20 topics), however the article length (average 317) is twice shorter than our datasets. Applying our Multinomial NB on this dataset yields around 75% accuracy. In Table 3, we summarize several state-of-the-art text classification methods and their accuracies. We should notice that the datasets in these articles has less labels and shorter article length than our datasets.

Table 3: State-of-the-Art Text Classification Accuracies

Name	Method	Dataset	# Labels	Accuracy %
Yoon Kim (2014)[1]	CNN-static	TREC	6	92.5
Zichao Yang et al. (2016)[2]	HN-ATT	Yelp' 15	5	71.0
Duyu Tang et al. (2015)[3]	LSTM-GRNN	Yelp' 15	5	67.6
Alexis Conneau et al. (2016)[4]	VDCNN	Yahoo!Answer	10	73.4
Xiang Zhang et al. (2016)[5]	ngrams	Yahoo!Answer	10	68.5
Stanford Classifier[6]	-	20Newsgroups	20	88

4.2 Deployment

Now we characterize the 150,000-document unlabelled deployment dataset.

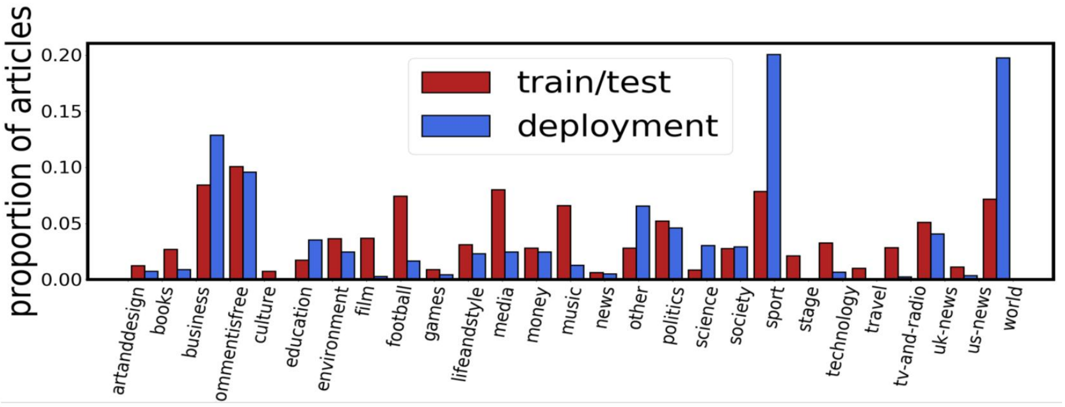

Figure 7: Normalized histograms showing the distribution of topics in the original 10000 training/test set that we trained our CNN on versus the distribution of topics predicted by our model.

Here, we can see that the distribution is not quite equal to the original distribution. Most importantly, it classified (20%) of the deployment corpus as sports articles, which seems excessive and somewhat uncharacteristic of a news source like the *Guardian*. We attribute this to the fact that assuming that articles coming from the same source have the same distribution is likely incorrect.

5 Conclusion/Future Work

In this report, we demonstrate the superior application of CNNs against sequence models and Naive Bayes in analyzing and labeling document topics in digital news media. We find that for this task, CNNs outperform classical and state-of-the-art sequence models, achieving 85%-87% accuracy, performances which we believe are comparable to many literature results on much simpler classification problems.

For future considerations, we consider four proposals. If we had more computational resources, ensemble models would be interesting to investigate. this would partially solve the large document problem by splitting the documents into smaller sub-documents, which are then clustered into mini-corpora which are trained on with separate algorithm. At the end, each algorithm votes on the topic classification and the majority is taken to be the final classification. Additionally, attention mechanisms have also been explored in CNNs, particularly for image captioning. We also believe that a rigorous implementation of attention in CNNs can also be used to perform attention in different locations of long documents. Given the vast amount of unlabelled data, another proposal would be to use semi-supervised learning techniques, which are used exactly for such situations to exploit some of the unlabelled data combined with the labelled data to infer insights on the classes that the unlabelled body of data might have.

6 Contributions

Nathan Zhao works on Data preprocessing/analysis and Naïve Bayes. Jiahui Wang works on Convolutional Neural Net. Xianling Long works on Recurrent Neural Net. We would like to acknowledge Frederic Filloux for providing the *Guardian* and the *Economist* datasets.

The Github repo is available at `https://github.com/zhaonat/CS230-Deep-Learning-NLP.git`.

References

[1] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[2] 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 American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, 2016.

[3] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1422–1432, 2015.

[4] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, vol. 1, pp. 1107–1116, 2017.

[5] X. Zhang and Y. LeCun, "Text understanding from scratch," *arXiv preprint arXiv:1502.01710*, 2015.

[6] Stanford NLP group, "Stanford classifier," *https://nlp.stanford.edu/wiki/Software/Classifier/20_Newsgroups*.

[7] D. Yogatama, C. Dyer, W. Ling, and P. Blunsom, "Generative and discriminative text classification with recurrent neural networks," *arXiv preprint arXiv:1703.01898*, 2017.

[8] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of cnn and rnn for natural language processing," *arXiv preprint arXiv:1702.01923*, 2017.

[9] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International Conference on Machine Learning*, pp. 2048–2057, 2015.

[10] F. Chollet *et al.*, "Keras." `https://github.com/keras-team/keras`, 2015.

[11] 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, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.