A Study on LSTM and CNN Models for News Classification Using Word2Vec Embedding

Eren Aldis[1], Elif Bilgin[2], and Juhi Malani[2]

[1]Department of Applied Mathematics and Statistics, Johns Hopkins University
[2]Department of Computer Science, Johns Hopkins University

Abstract

In this paper, we explore the potential of using CNNs and LSTMs to solve the problem of classification of news headlines into four categories - business, entertainment, health, science and technology. We show that pretrained word2vec embeddings from the Gensim module [8] are able to improve on pure deep architectures. We present the preprocessing steps taken in our approach as well as methods used, and conclude with the presentation of our results for both architectures. We demonstrate using cross-validation that LSTMs are able to outperform CNNs on this particular text classification task. We also investigate the performance of both architectures in a more complex classification problem and conclude that our results are statistically consistent.

1 Introduction

With smart phones and smart devices growing more popular by the day, it is no surprise that humankind has experienced a dramatic decrease in attention span. As our ability to focus on things such as a news article has been compromised severely in the recent years, retrieval of information that is fast and accurate has become invaluable. Therefore, in the case of keeping up with events from around the globe, we nowadays want concise and relevant information, and we want it fast.

These facts have played a key role in our project selection, as we have chosen to classify news articles based on their headlines in one of the four following classes: Business, Health, Science and Technology, and Entertainment. Deep Learning tools have gained significance for this task over recent years, with platforms such as Quora and Medium using a similar classification technologies to filter the articles according to user preferences.

Traditional machine learning techniques such as Logistic Regression and Multinomial Naive Bayes have been reported to yield high accuracies for simpler text classification tasks, but with improvements in deep learning methods and computation speed, an increased focus has been on network architectures such as LSTM and CNNs [4]. For the scope of our project we have built Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) models, and compared their performances for our text classification problem.

2 Dataset

We have utilized the News Aggregator Dataset from the UCI Machine Learning repository, which is an open source repository with over 300 data sets that can be used by the machine learning community. The dataset itself consists of news article headlines, URL links and the four categories for 422,937 news articles that were collected online in 2014. These four categories are business, technology, entertainment and health, as denoted by from here on out as B, T, E and M respectively.

We also investigate a Huffington Post dataset with news headlines collected for the years 2012 to 2018 [7]. The dataset contains 200,000 news headlines and 41 news categories with classes ranging from politics to weddings. Although most of the dataset is comprised of politics, wellness and entertainment categories (32k, 17k, 16k respectively), the rest of the categories are well balanced with an average of around 3000 entries per category. Thus, this is a much harder dataset to run the classification algorithms on.

3 Preprocessing

To begin, we started with preprocessing the dataset into an efficient and palatable version of itself. For best results, the steps follow the same vein as is used for current NLP programs. The initial dataset has more than 400,000 newspapaer headlines from 2014 sorted into 4 major categories, business,science and technology, health, and entertainment, labelled b,t,m,e respectively Each in-

stance has values for the following columns. We were mainly concerned with the Title and Category columns for our specific task.

Real world data almost always initially presents itself as a mess due to human error, unconventional results, unavailable information, or a combination of these aforementioned factors. Therefore, multiple methods were employed to account for this messiness.

To start off, the datasets were balanced by the number of categories. The baseline was the category with the least entries. Thus, from each category this baseline amount was sampled to create a balanced set of data. However, the unbalanced dataset was also kept to compare against models trained on the balanced dataset. As balancing the datasets lead to a signficant loss in training data, this could and has resulted in decreased accuracy. For data cleaning, abnormalities and left-behind components in the news headlines, specifically in the form of URLs were tracked and removed.

Finally, text to word sequencing was used to clean the remaining data by removing all punctuation and unecessary characters before stripping it into a vector indexing individual words.

Traditionally labelled categories (simple conversion into integers) would lead the algorithm to assume higher values to be "better" leading to inaccurate output. This error was mitigated with a combination of *text_to_word* sequence for the news headlines and one-hot encoding for the classes resulting in a dataset sorted by categorical values, as opposed to ordered indices.

Lastly, all the vectorized news headlines were padded for uniform input to ease the training process.

The models were then trained using the 50,000 most commonly occurring words in the word dictionary, as including the rest of the words did not yield to improved results.

3.1 Word2Vec Embedding

Commonly used in natural language processing, Word2Vec is a predefined function to map words into vectors of numeric values to to be used as an additional hidden layer in the network.

We have used the Common Bag of Words(CBOW) approach to compute frequency of words in the text corpus and maps them into a dense token. The words are assigned values with a particular goal in mind, the relation between words can be calculated by the difference between their values. The smaller the difference, the greater the correlation between them.

Used as an additional parameter,this aids word representation by leveraging their context, *ie. categories* by "*weighing*" sorted words by their relative surroundings. Another immense advantage, of using this approach is that when individual word values are calculated by averaging over each occurrence, they can then relate words with indirect correlations with a greater accuracy, making networks, both, cost effective and accurate.

As getting a decent word2vec model would require large set of data and significant computational power, we used the pre-trained weights downloaded through Gensim [8].

4 Models

In the past, researchers have attempted to solve the problem using basic classifiers such as a Multinomial Naive Bayes classifier as well as Logistic Regression to reach 92% and 94% test accuracy, respectively[]. Our general approach has been to attempt to beat these accuracies using deep learning models that we have been exposed to in lecture, namely LSTMs and CNNs. We initially built a simple multi-class (one-vs.-rest) logistic regression model to provide us with a baseline accuracy.

4.1 Logistic Regression

In the Kaggle news classification competition with the same News Aggregator Dataset [6], logistic regression reached the highest accuracy among the traditional machine learning models. Thus, we decided to take the baseline accuracy to be the validation accuracy from the logistic regression model to benchmark the deeper architectures against.

Logistic regression, along with more traditional methods, fails to capture nonlinear relationships in high dimensional subspaces - particularly when there are multiple nonlinear decision boundaries. Since we had sufficient data to train deeper architectures, we hypothesized that deep learning models could improve the accuracy for our classification problem. Previous literature shows that for text classification problems, CNNs and LSTMs are the most popular choices for possible deep architectures[5].

4.2 CNN

We have followed the simpler CNN architecture proposed in Kim [1] for text classification, comprising of an input layer, followed by a single convolution and max-pooling layers with rectified linear unit (ReLU) activation, and a densely connected final layer with a soft-max activation function. We also consider a more complex model with multiple convolutional layers.

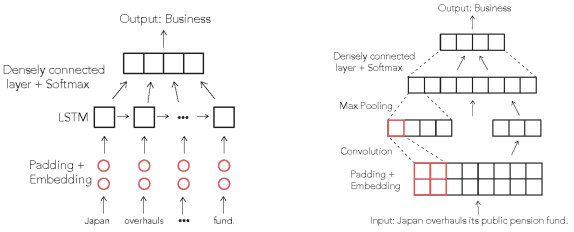
Using fixed sized sequential inputs padded by the maximum length d, (which was only 19 in this

case), the convolutions are able to extract local information from the word sequence to be used in the feed-forward process. Locality is captured through convolving $k \leq d$ sequential components of the words together. This method, however, possibly discards valuable holistic information embedded in the sentence structure. Nonetheless, [1] show that convolutional structures still are able to outperform traditional machine learning approaches. Moreover, one of the main advantages of using a convolutional approach is that it significantly decreases the training speed.

4.3 LSTM

First introduced to resolve the vanishing gradient problem with Recurrent Neural Networks (RNNs), LSTMs [2], have proven effective in many natural language processing tasks [3]. Unlike CNNs, we expect LSTMs to approach the sequential structure of the word embeddings holistically to capture "long-distance" relationships between words in the news headlines. Our LSTM model is comprised of a single LSTM layer followed by a densely connected layer with softmax activation.

We also consider a joint model using a convolutional layer followed by a LSTM layer, followed by a densely connected layer with softmax activation proposed in [4], which has been noted to yield improved results for speech recognition tasks. However, it is possible that this might in fact slow down the network and still not be able to capture the "long-distance" information in the sentence structures as the first layer is a convolutional layer.



(a) LSTM model **(b)** CNN model

Figure 1: The two architectures with embeddings

4.4. Parameter Tuning and Validation

Through testing with different loss functions and optimizers, the adam optimizer and categorical loss function performed the best and thus, was used for each of the models. We also inserted dropout layers between every layer in all architectures (except btw. convolution and max-pooling, and after input layers) to prevent overfitting, with parameters around 0.5, the networks returned the highest validation accuracies.

We also note that training on balanced classes returned lower validation accuracies for all of the models, thus the results below are given from training using the full unbalanced data (after initial preprocessing and data cleaning).

Lastly, the results were evaluated on the validation set using 5-fold cross validation, that is 20% of the data was being used for validation and the rest used for training at each of 5 model evaluations. The validation accuracies given below are the mean accuracies over the 5 cross validation iterations. We expect this method to yield a more robust comparison between the models, by giving a p-value for the significance of the results.

5 Results

We compare the models explained above for the news classification problem, using the 4-class News Aggregator Dataset. In the table below, we give the cross-validated validation accuracy and validation loss results after training each network for 5 epochs, which through observation was deemed to be the optimal amount. In the results below, we also illustrate the differences between training the models using the pre-trained word2vec embeddings and without.

Model	Acc	p-value
Logistic Regression	94.83	-
CNN (Single Conv.)	94.98	< 0.10
CNN (Triple Conv.)	87.19	$< 10^{-6}$
LSTM	95.02	< 0.02
CNN + LSTM	90.18	$< 10^{-6}$
CNN w/ word2vec	95.01	< 0.02
LSTM w/ word2vec	**95.24**	< 0.01

Table 1: Model performance on the News Aggregator Dataset (Results averaged over Cross Validation), p-values computed against Logistic Regression validation accuracy using Wilcoxon Ranked-Sum test

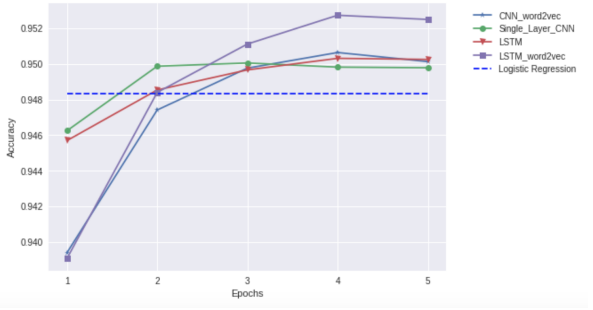


Figure 2: Comparison of mean validation accuracies for models on the News Aggregator Dataset

The CNN with 3 convolutional layers (CNN - Triple Conv.) and the CNN with an LSTM layer on

top (CNN + LSTM) models achieved sub-par performance with respect to the traditional logistic regression model. On the other hand, the LSTM and CNN models achieve better performance than the traditional model, especially when word2vec embedding is used. However, it should be noted that the increase in the accuracy for the CNN after using word2vec embeddings is not significant (p-value > 0.2), whereas for LSTMs this difference seems to be significant (p-value < 0.01).

We give the confusion matrices for the CNN and LSTM models after the word2vec embedding is used in the tables below.

		Predicted			
		Tech	Ent.	Health	Bus.
Actual	Tech	27180	390	262	1248
	Ent.	349	37285	138	311
	Health	311	196	10686	131
	Bus.	1304	287	111	25411

Table 2: Confusion matrix for LSTM (word2vec)

		Predicted			
		Tech	Ent.	Health	Bus.
Actual	Tech	27044	389	364	1283
	Ent.	394	37196	182	311
	Health	274	200	10707	143
	Bus.	1289	326	140	25358

Table 3: Confusion matrix for CNN (word2vec)

The CNN and the LSTM classifiers that use the word2vec embeddings yield close accuracies. We also observe that even after longer training times and further tuning this relationship stays the same. From this we infer that this might be due to the simplicity of the problem or the 4-classs dataset. Therefore, we compare these two models against each other on a harder news classification problem with 41-classes using the Huffington Post dataset. We give the results below in Table 4.

Model	Acc	Loss
CNN w/ word2vec	57.72	1.533
LSTM w/ word2vec	**60.74**	1.397
p-value	< 0.01	< 0.01

Table 4: Model performance on the Huffington Post Dataset (Results averaged over Cross Validation), p-value computed to test significance between validation accuracies of two models using Wilcoxon Ranked-Sum test.

The models were both trained with and without balancing the classes and again the latter model performed better. The output from other models are not noted here as the CNN and LSTM models trained using the word2vec embedded data performed better on the previous News Aggregator Dataset.

The results above show that after embedding the input with word2vec pretrained weights, LSTM is able to perform significantly better than the CNN model. This significance in performance is more evident for the more complex Huffington Post dataset. The reason the overall performances are relatively low is due to the lack of data for the 41 categories, for which some have only around 1000 entries.

6 Discussion and Conclusions

The results in section 5 show, (1) the word2vec embedding yields to an improved prediction accuracy, and (2) LSTMs combined with word2vec embeddings outperform CNNs in the news classification task for our two datasets. We have discussed reasons for (1) in Section 3: Preprocessing and for (2) in Section 4: Models.

Now we will discuss the possible shortcomings of our approach. For this we should note what our results do not illustrate. First, our results do not show that LSTMs are better than CNNs for text classification. There could be (and should be by free lunch theorem) text data for which CNNs could yield a better classification performance than LSTMs. Possibly a dataset where local information embedded within the sentences could yield such a result. Our comparison, thus, is only an evaluation of the two models on a news headline classification task.

Second, our results, do not show with absolute certainty that LSTMs are consistently better than CNNs for news headline classification tasks. Such a result would require the comparison of the models on a much larger amount of datasets, which would thus require a significant computation power to train and human power to tune. However, our results along with the current literature do give a good indication of the performance difference between the two models.

Lastly, by the universal approximation theorem, with enough data and enough parameters, we would expect these two models to yield similar performances with the right tuning. Nonetheless since we are working with finite data and perhaps insufficient data in the case of the Huffington post dataset, the theorem might not be too relevant for the subject matter. However, it raises another issue that is, with limited time for training at hand, our search for better hyperparameters and parameters, were limited. For hyperparameter selection, we have evaluated more complex CNN models as well as simpler ones, and observed that the simpler

ones performed better. That is why we used the CNN with a single convolutional layer in most of our analysis. Yet, it could be argued that a better search process could have yielded a better hyper-parameter and parameter selection to result in a higher validation accuracy than the LSTM model. This is a common shortcoming for many papers written in the subject and this is unfortunately only a fact we must accept.

Finally, it must be noted that although LSTMs were significantly better than CNNs in the news headline classification tasks for our data, CNNs were significantly quicker. Therefore, if the validation accuracies are very close, one might choose to opt for using CNNs to train larger datasets rather than LSTMs. However, if the accuracy is of vital importance the LSTMs seem to be the right choice.

References

[1] Yoon, Kim. *"Convolutional neural networks for sentence classification".* In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pg 17461751. 2014.

[2] Sepp Hochreiter and Jurgen Schmidhuber. *Long short-term memory.* 1997. Neural Computation, 9(8):17351780.

[3] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. *Long short-term Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval.* 2016. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 24(4):694707.

[4] Sainath, Tara N, Et al. *Convolutional, long short-term memory, fully connected deep neural networks.* Google, Inc., New York, NY, USA.

[5] Guggilla, Chinnappa et al. *CNN- and LSTM-based Claim Classification in Online User Comments.* COLING (2016).

[6] Luis Bronchal.

https://www.kaggle.com/lbronchal/classifying-with-logistic-regression-0-9473 Classifying with Logistic Regression

[7] Rishabh Misra.

https://www.kaggle.com/rmisra/news-category-dataset/home Huffington Post Dataset.

[8] Radim Řehůřek and Petr Sojka. *Software Framework for Topic Modelling with Large Corpora* Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, 45–50, 2010. Google, Inc., New York, NY, USA.

Mikolov, Tomas Chen, Kai Corrado, G.s Dean, Jeffrey. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of Workshop at ICLR. 2013.

Liu, Haixia. (2017). Sentiment Analysis of Citations Using Word2vec

Mikolov, Tomas et al. Distributed Representations of Words and Phrases and their Compositionality. NIPS (2013).