News Classification with Bidirectional Long Short Term Memory (Bi-LSTM) with WandB

DI504 – Foundations of Deep Learning Term Project Report

Mert Mecit
Information Systems Department
Middle East Technical University
Ankara, Turkey
Email: mmecit@metu.edu.tr

Abstract— **In this paper, I tried to build a bidirectional LSTM model to predict classes for news. I utilized a widely used dataset known as AG News. First of all, a descriptive statistics analysis is performed to understand the structure of data and have an idea about further preprocessing steps. After ensuring the balance in data and there are no null values, I proceeded to do the preprocessing step. In this step, I applied tokenization and padding into my dataset for my model to perform better. My model is a bi-directional LSTM model with different types of layers. In its first hidden layer, it has an embedding layer. Then, it has layers such as BiLSTM, max-pooling layer, fully connected layers, and dropout layers. After fine-tuning my parameters, I started the training. Training is done and weights and biases are calculated via WandB. Overall, the model has an accuracy of 91%, which is slightly below the average in the literature. Some limitations of the study include computational capacities, simplicity of the model, and dataset shortage in the literature.**

Keywords-classification; LSTM; BiLSTM; AG News; WandB;

I. INTRODUCTION

Text classification is an important task in both machine learning and deep learning domains. The main concern in text classification is assigning predefined categories to given texts. News classification is a subset of text classification mainly dealing with news data. It has become particularly important in the era of big data as now people have an abundance of news from thousands of resources, it is crucial to correctly categorize news beforehand to save a lot of time while searching.

LSTM is a widely used algorithm in the natural language processing (NLP) domain and will constitute our main model in this paper. It is a type of recurrent neural network (RNN). The main difference between LSTM and other main algorithms is it has feedback connections and it does not have to use single data points, it may use sequential data (like text) as well.

Weights & Biases or wandb is the machine learning platform for developers to build better models faster. Use W&B's lightweight, interoperable tools to quickly track experiments, version and iterate on datasets, evaluate model performance, reproduce models, visualize results, spot regressions, and share findings with colleagues.

Here, first, a brief literature review will be presented. Then, I will introduce my dataset and provide a descriptive analysis of the variables in my dataset. Then, first a preprocessing to the dataset will be done and the model will be constructed. Finally, performance will be evaluated and compared with the other models created on a similar dataset that I can find online.

II. LITERATURE REVIEW

LSTM is widely utilized in the NLP area and our case, text classification. In their paper, Liu, and Guo [1] proposed a new LSTM model, which utilizes a bidirectional LSTM model which they call “attention-based bidirectional LSTM (AC-BiLSTM) and they show that this model outperforms other widely used text classification models in the deep learning domain. Basiri et al. [2] also utilized a new LSTM model that is based on an attention-based approach. They also used bidirectional LSTM and GRU layers and used this model on sentiment data. Their model outperformed other classification methods on this subject. In their paper, Nowak, Scherer, and Taspinar [3] tried to use 3 different LSTM algorithms (namely LSTM, B-LSTM, and GRU) on Amazon’s comment database. Their algorithms were mainly aimed to classify comments either positive or negative and improved the accuracy of the best machine learning algorithm solution of that time by 15%. In their paper, Jang, Kim, Harerimana, Kang, and Kim [4], tried to combine a CNN and LSTM approach into text classification. They also compared their results with each of them individually. They used a popular dataset, the IMDB comment dataset. They found that their hybrid model yields better results in classifying comments. In their paper, Nergiz, Safali, Avaroglu, and Erdogan [5] tried to classify Turkish news by LSTM using the Fasttext model. They categorized news into

seven categories and tried three models with Fasttext, Word2vex and, Doc2vec. Fasttext yielded the best results with an accuracy of more than %96.

III. DATASET

a) Dataset

In this part, I will explain the dataset I am using. This dataset is called AG News. This dataset is mainly used for classifying news based on its title and content. It classifies data into 4 categories:

1. World
2. Sports
3. Business
4. Sci/Tech.

It has three columns, class, title, and description (content). This dataset originally consists of more than 1 million news from more than 2000 resources and took 1 year to complete. Dataset is constructed by Zhang, Zhano, and LeCun. [6]

An example from the dataset looks like this:

Index	Title	Description
3	Oil prices soar to an all-time record, posing new…	AFP – Tearaway world oil prices, toppling record…

Figure 1. Data Example

b) Descriptive Statistics

Before analysis, we should always check the descriptive statistics of the variables in our dataset. Before exploring it, I should mention that, for my analysis, I have combined title and description into one variable to save time and called it "Description". When we check the descriptive statistics of the variable index and value counts of the "description" value:

	Index	Description
Count	120000	120000
Mean	2.5	37.85
Std	1.12	10.08
Min	1	4
25%	1.75	32
50%	2.5	37
75%	3.25	43
Max	4	177

Figure 2. Descriptive Statistics

	1	2	3	4
Count	30000	30000	30000	30000

Figure 3. Index Counts

Descriptive statistics here tells us important things here. First of all, these are the descriptive statistics for our train dataset. It consists of 120000 rows, which is good because, in deep learning models, we generally want to use a high amount of data.

One important insight we can get from the table below is, all of our indexes have the same amount of data. This is crucial as if there were to be a significant imbalance, I needed to apply different methods that work well with imbalanced data or somehow (for example via augmentation) I would need to balance my data to proceed.

The last thing I want to mention about the descriptive statistics is in the description part. We can see here, on average, I have 37 words in the description part. This can affect our runtimes significantly and compared to other datasets, we can say this is fairly low.

c) Null Values

Last but not least, null values may affect the performance. However, luckily, as this dataset is a refined one from the original one, which has millions of data, we do not have any null values.

IV. PREPROCESSING

Preprocessing is generally done for having proper data, such as eliminating duplicates, imputing null values, or correcting data types. However, in the NLP case, the main consideration is the performance and cost of the model. As we work with huge data in the deep learning domain, we need to somewhat shorten our text data. Two widely used methods are utilized in my model.

a) Tokenization

Tokenization is the process of breaking a whole text into smaller parts. It can be done in several ways. One can turn a paragraph into sentences, sentences into words, or words into letters. In our case, it is sentences into words. Also, punctuations are removed from the sentence. So,

"Hello, I am Mert!"

becomes

"Hello", "I", "am", "Mert"

b) Padding

To work properly, in a deep learning model, every input has to be in a similar shape and similar size. Padding is crucial in this case as it provided this. You can be sure that your inputs have a similar size. I padded them into their max values.

V. MODEL

As I have mentioned, my model will be based on a bidirectional LSTM architecture. Each of the layers, loss function, optimizer, etc. will be explained in detail. Please note that figure 4 is just given for understanding the overall architecture. It does not contain feedback arrows of a normal LSTM model. In terms of hyperparameters, unfortunately, I

could not use hyperparameter optimization libraries as epochs were taking a lot of time and my computer was freezing while trying 5-10 different hyperparameters at the same time. So, I needed to check them one by one.

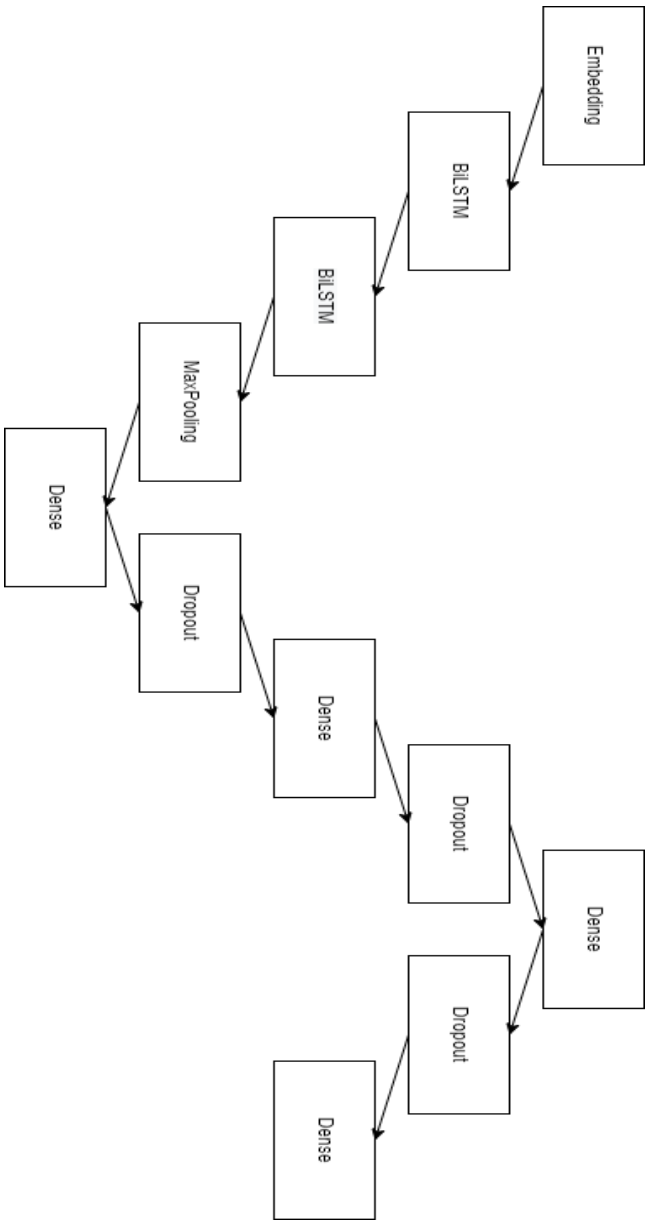

Figure 5. Model

a. Layers

i. Embedding Layer

The embedding layer here works as the first hidden layer in my network. I initialized them with random weights, so it can learn word embeddings. I initialized it with an input length of maximum length in my dataset, an input dimension of 10000 (totally random), and an output dimension of 32.

ii. Bidirectional LSTM Layers

The model consists of two bidirectional LSTM layers. They duplicate the first recurrent layer and create two layers side-by-side. It makes an input to the first layer and has a copy of reversed input for the second one. In our model, the first one has an input size of 128 and the second one has an input size of 68.

iii. Max Pooling Layer

A max-pooling layer is implemented to downsample the feature maps. For the text classification domain, a special type of max pooling, which is called global max pooling is widely preferred. As they mention in their work, Lin, Chen, and Yan [7] mention that "The idea is to generate one feature map for each corresponding category of the classification task in the last mlpconv layer. Instead of adding fully connected layers on top of the feature maps, we take the average of each feature map, and the resulting vector is fed directly into the softmax layer. One advantage of global average pooling over the fully connected layers is that it is more native to the convolution structure by enforcing correspondences between feature maps and categories."

iv. Dense Layer

The dense layer is also known as the fully connected layer. In those layers, the neuron applies a linear transformation to inputs with weights. In my model, I have 8 fully connected layers. All of them have linear activation functions (no activation is applied). The last one is the final layer of my network and has 4 as the dimension of the output space (as we have 4 categories).

v. Dropout Layer

Dropout layers are good ways of preventing overfit in neural networks. They regularize the network as they tend to prevent weights to get very high. In my network, I have 7 dropout layers, each of them dropping 0.5 of the dataset.

b. Other Considerations

i. Final Activation Function

Of course, the layers and their types were not the only decisions I needed to make. One of the crucial considerations while building the network is to decide the final output function. In my case, it is a softmax function. It normalizes outputs to a probability distribution and then finally picks the most probable one as the final output. In my case, it can guess either one of the four news categories in my dataset.

ii. Optimizer

The optimizer in my model is Adam. I have tried 3 different optimizers: SGD, Adam, and Adadelta. Adam is

chosen by trial and error. They are used under 4 different cases and their average accuracies are given in figure 6.

	SGD	**Adam**	**Adadelta**
Average Accuracy	0.79	0.87	0.83

Figure 6. Optimization Selection

iii. Loss Function

The purpose of a loss function is to compute what the model should be minimizing during the training phase. In my case, it is sparse categorical cross-entropy. Again, I have tried 3 of them, namely sparse categorical cross-entropy, categorical cross-entropy, and KLDivergence. Sparse CCE came out to be superior and used in the final model. For the other ones, I needed to do a one-hot representation as well. Figure 7 summarizes average accuracies

	Sparse CCE	**KLDivergence**	**CCE**
Average Accuracy	0.86	0.66	0.75

Figure 7. Loss Function Selection

iv. Early Stopping Mechanism

An early stopping mechanism is implemented in our model to save time. The training process is very time-consuming and we want to stop the training process when our model cannot improve (or improve less than a specified amount in our case). In my case, I tried four different early stopping deltas and those are (figure 8) the results where they stop and when they stop. In terms of trade-off, I believe, the 1e-3 difference gives the best result.

	1-e3	**1-e2**	**1-e1**
Stopping Accuracy	0.88	0.89	0.91
The Time it Takes (in mins)	79	121	247

Figure 8. Stopping Mechanism Selection

v. Batch Size

Again, the batch size is an important consideration as it has a great trade-off between the quality of the model and time. I tried 3 batch sizes with huge differences between them and I needed to use a batch size of 1024 (generally a power of two is used) as it makes life easier in terms of time. Figure 9 summarizes this concern.

	64	**512**	**1024**
Average Accuracy	0.85	0.86	0.89
Time it takes	206	149	84

Figure 9. Batch Size Selection

vi. Epochs

The number of epochs was the most problematic part of my analysis as it affects the training process drastically. Although it depends on other variables as well, one epoch of training was approximately taking 30 minutes. The stopping mechanism was a lifesaver in this situation but still, if I were to use more than 5 epochs, it would take forever to finish the training process. For each of the decisions I talked about previously, tests were made with 1 epoch, then their results are compared. However, when all decisions are made, the final model was built with 10 epochs.

vii. Weights & Biases

In my model, the best weights and biases are built with the wandb library of Python. They are provided in the zip file as well.

viii. Train, Validation & Test Split

Our dataset was already divided into training and test split. The training set consists of 120 thousand news and their classes and training set consists of 8 thousand news. I divided my training set into 90% training and 10% validation, so I used 10-fold cross-validation.

VI. Results & Comments

With our hyperparameters in mind, I have finally done training my model with 10 epochs. It took more than 1 hour to finish it thanks to my early stopping mechanism and at the end of the second epoch, my run has ended. All of the weights are calculated with wandb methods. Overall, my accuracy on my test data is 0.91. I think it is a fairly satisfying result as my computer's computational capabilities were limited and I could not conduct all of the analysis with all of the different combinations I want. Here, I will summarize how my model compares to other models that are in the literature. This table is from Jiang et al.[8]

Model	Accuracy on AG's News
Char-CNN	0.89
Char-CNN (Conceptualized)	0.91
BERT	0.93
BERT (Conceptualized)	0.94
BiLSTM (My Model)	**0.91**

Figure 10. Model Comparison

Although my model is outperformed by the literature, I believe my model was the simplest one and this slightly lower accuracy may be tolerated by its running time.

Also, it is worth examining the classes individually as we can notice in which class my model performs better than the others.

	Predicted Label				
True Label		1	2	3	4
	1	0.93	0.03	0.03	0.01
	2	0.06	0.94	0.01	0.00
	3	0.07	0.01	0.87	0.05
	4	0.09	0.01	0.10	0.80

Figure 11. True and Predicted Labels

In world and sports news, the model performs better. Especially in sci/tech news, it performs with an 80% accuracy. However, this can be expected as to whether a sci/tech news is related to business or world is generally cannot be understood easily. Also, sports news are the ones that are barely falsely predicted and this is logical as we generally tend to not confuse sports news with business or sci/tech. Overall, accuracy, precision, and recall values are given in figure 11.

Recall	Precision	Accuracy
0.91	0.91	0.91

Figure 11. Recall, Precision, and Accuracy

Recall or correctly predicted positive observations to all observations in the class, precision or positive observations on total positive predictions and accuracy, a weighted average of recall and precision all come out to be 0.91.

Finally, I will show some examples that I have provided into the system and how my model labeled them.

News	Prediction
Eminem Terrified As Daughter Begins Dating Man Raised On His Music	World
7 Ways to Make Money While Waiting for Disability Benefits	Business
10 Signs That You Will NOT Make It As A Successful Photographer	World
Winter Olympics quiz: Test your knowledge of the Games	Sports

Figure 12. Examples

Although the difference between some categories is not crystal clear, I think the first, second, and last ones are definitely correctly labeled. However, the third one is ambiguous. It can be world, business, or even sci/tech to some extent.

VII. Conclusions & Limitations of the Study

Throughout the paper, I tried to build an LSTM model for predicting text classification or specifically news classification. For this purpose, the AG News dataset is utilized.

First of all, a descriptive statistics analysis is applied to ensure data is ready for analysis. After that step, preprocessing steps for natural language processing work are done with tokenization and padding.

My model has many different types of layers including embedding layer, bidirectional LSTM layer, pooling layer (a global max one), fully connected layers, and dropout layers. Then, I fine-tuned my hyperparameters with trial and error methods. Optimizer, loss function, stopping mechanism, the batch size is all selected with numerous trial and errors.

After the training phase is done, my model yielded an accuracy of 0.91, which is slightly below the models proposed in the literature but has some benefits:

i. It is a very simple model which can be easily understood.
ii. As it is simple, the training phase is relatively shorter.

I will conclude my work with some suggestions I can make for further research as there are many limitations I have experienced throughout this paper:

i. Although a simple model has the benefits I have talked about above, my model might be oversimplified. Some additional layers or parameters may be added to enhance the performance even further.
ii. This model should be run on a high-performing computing device. This was the most critical part for me as I did not have a high-performance computer, I could not fine-tune my hyperparameters with methods such as grid search, random search, or Bayesian search. With those methods, the model would be much more efficient.
iii. Unfortunately, I could not find another news dataset that I can use on my model. It would make my model much more reliable.

VIII. References

[1] Gang, L., & Jiabao, G. (2019). Bidirectional LSTM with attention mechanism and convolutional layer for text classification, *337*, 325–338. https://doi.org/10.1016/j.neucom.2019.01.078

[2] Basiri, M. E., Nemati, S., Abdar, M., Cambria, E., & Acharya, R. U. (2021). ABCDM: An Attention-based Bidirectional CNN-RNN Deep Model for sentiment analysis. *FUTURE GENERATION COMPUTER SYSTEMS-THE INTERNATIONAL JOURNAL OF ESCIENCE*, *115*, 274–294. https://doi.org/10.1016/j.future.2020.08.005

[3] Nowak, J., Taspinar, A., & Schrerer, R. (2017). International Conference on Artificial Intelligence and Soft Computing. In LSTM Recurrent Neural Networks for Short Text and Sentiment Classification.

[4] Jang, B., Kim, M., Harerimana, G., Kang, S.-ug, & Kim, J. W. (2020). Bi-LSTM Model to Increase Accuracy in Text Classification: Combining Word2vec CNN and Attention Mechanism.

[5] Nergiz, G., Safali, Y., Avaroglu, E., & Erdogan, S. (2019). 2019 International Artificial Intelligence and Data Processing Symposium (IDAP). In Classification of Turkish News Content by Deep Learning-Based LSTM Using Fasttext Model.

[6] Zhang, X., Zhao, J., & LeChun, Y. (2015, September 4). Character-level Convolutional Networks for Text Classification. arXiv. Retrieved from https://arxiv.org/abs/1509.01626

[7] M. Lin, Q. Chen, and S. Yan, "Network in network," arXiv:1312.4400, 2013.

[8] Jiang, H., Yang, D., Xiao, Y., & Wang, W. (2019). Understanding a bag of words by conceptual labeling with prior weights.