Leveraging LDA for Article Classification

Junhui Cho <jh00@bu.edu>, Jeong Yong Yang <chrisyan@bu.edu>

Abstract

The project aims to extend the principles of text classification to the realm of multi-class classification of news articles by genre or topic. Drawing from our previous experience in binary classification where we distinguished emails as spam or not spam, this initiative broadens the scope to categorize texts into multiple classes. However, simply broadening a binary classification into a multi-class classification seems boring. Hence, we chose to come up with our own topics for the articles. The core methodology involves employing Latent Dirichlet Allocation (LDA), a renowned approach in topic modeling, to generate labels for news articles. These LDA-derived labels serve as ground truth for our classification challenge. The significance of this project lies not only in its application to the genre classification of articles but also in its potential as a versatile framework applicable to a wide array of text classification tasks across different domains. This endeavor enhances our understanding of multi-class classification techniques and underscores the adaptability of machine learning workflows in diverse text analysis contexts.

Data Cleaning and Processing

In our methodology, we incorporated the title of each article into its content, premised on the rationale that titles frequently encapsulate pivotal keywords which can significantly influence the topic determination process for each article.

Regarding data preprocessing, we adhered to the following standardized protocols:

1. Conversion of all text to lowercase to maintain uniformity and negate case sensitivity.
2. Exclusion of location information typically situated at the commencement of articles, as it was deemed irrelevant to the topic modeling process.
3. Elimination of stopwords, which are generally considered as noise in text data.
4. Removal of numerical characters and punctuation marks to ensure the textual data contains only meaningful words.
5. Application of lemmatization to words, thereby converting them to their base or dictionary form, which is beneficial for reducing the complexity of the dataset.

Additionally, we introduced an "ID" column, serving as a unique identifier for each article. This inclusion aimed to facilitate efficient tracking and referencing of individual articles within the dataset.

LDA

We employed a comprehensive approach to preprocess news article data, utilizing two key techniques: TF-IDF and Gensim's dictionary-based preprocessing.

Firstly, we used the TF-IDF method to evaluate the importance of words across all documents. This technique helped in identifying and filtering out less significant words, ensuring that the focus remains on terms that are uniquely relevant to each article. By emphasizing words with higher TF-IDF scores, we effectively enhanced the distinction between different genres of articles.

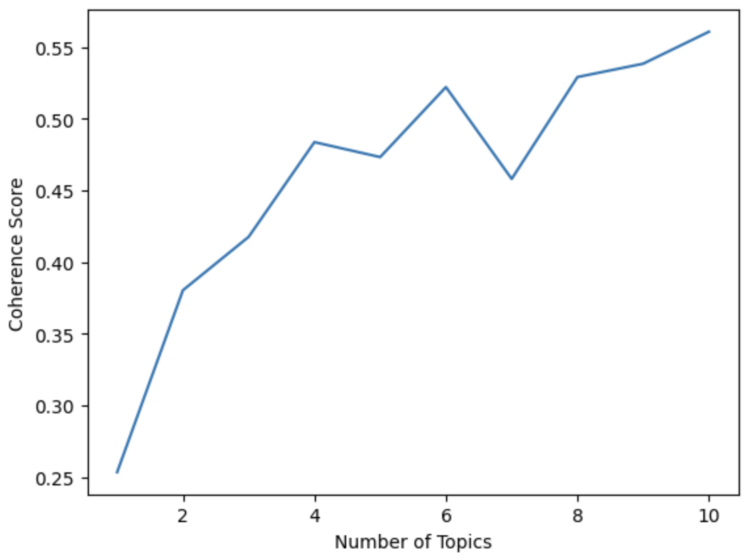
Secondly, we incorporated Gensim's dictionary capabilities, which are instrumental in analyzing document frequency for each word. This aspect of our preprocessing strategy involved filtering out extremes – both very common and rare words – based on their frequency of occurrence across all documents. By doing so, we refined our dataset, removing noise and irrelevant information.

Upon completing the conversion of our documents into a bag-of-words format, we proceeded to implement the LDA Model. This necessitated the determination of optimal hyperparameters to ensure the effectiveness of the LDA model.
The model encompasses three critical hyperparameters:

- The Number of Topics.
- The Dirichlet hyperparameter alpha, which influences the Document-Topic Density.
- The Dirichlet hyperparameter beta, which governs the Word-Topic Density.

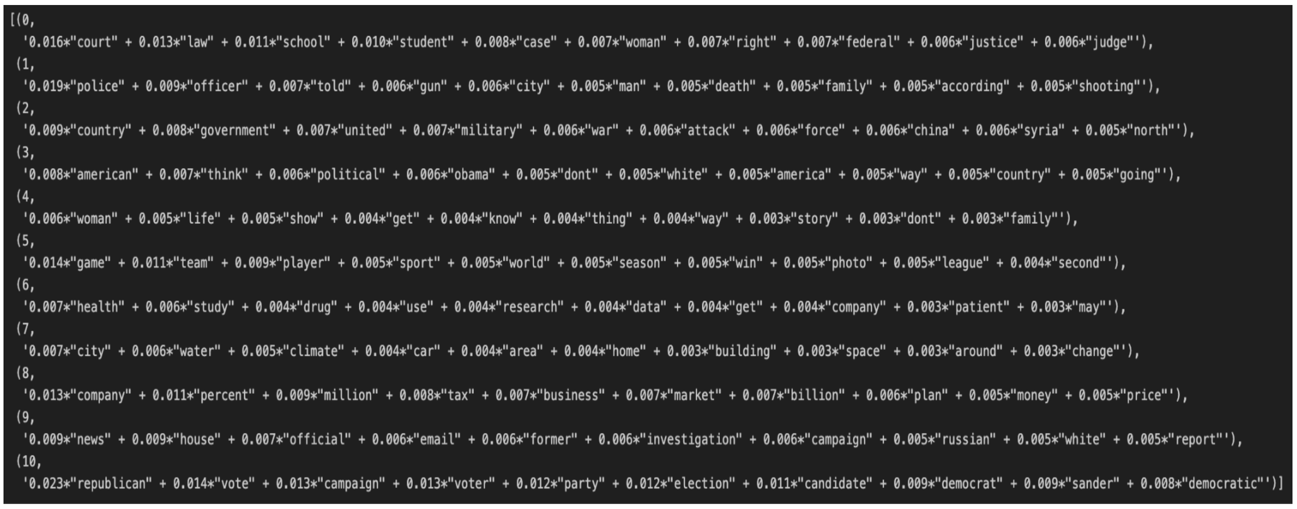
In our approach, we retained the default settings for the alpha and beta parameters. However, we conducted experiments varying the number of topics, recognizing its substantial impact on the delineation of topics within the corpus.

To assess the performance of the model, we employed the coherence score as a metric for evaluating the quality of the topics generated. The coherence score quantifies the semantic congruence amongst high-scoring words within a topic. Although multiple methodologies exist for computing coherence scores, our analysis adhered to the most prevalent and universally accepted method: the C_v score. It is generally acknowledged that a coherence score exceeding 0.5 is indicative of satisfactory topic quality.



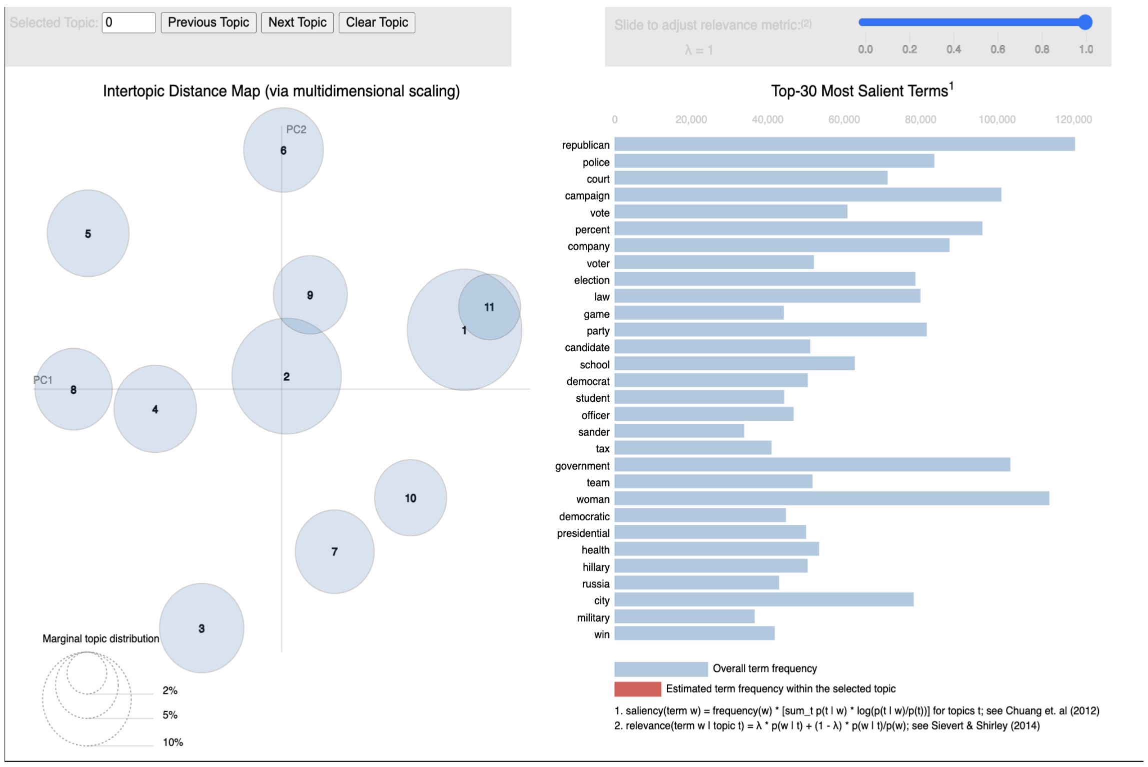
We carefully selected eleven as the optimal number of topics for our LDA model. The C_v score for eleven was around 0.55, which is good, and we aimed to strike a balance between coarseness and coherence. With fewer topics, we observed an oversimplification and a loss of variant topics. Conversely, a higher topic count resulted in overly fragmented, less interpretable, and overlapping categories. Eleven topics emerged as the most effective number, providing a comprehensive yet distinct categorization of our dataset, ensuring each topic was sufficiently specific yet broad enough to encapsulate the key themes inherent in our corpus.

The output of our LDA model with eleven topics is presented as follows, with each topic characterized by its most significant keywords:



```
[(0,
  '0.016*"court" + 0.013*"law" + 0.011*"school" + 0.010*"student" + 0.008*"case" + 0.007*"woman" + 0.007*"right" + 0.007*"federal" + 0.006*"justice" + 0.006*"judge"'),
 (1,
  '0.019*"police" + 0.009*"officer" + 0.007*"told" + 0.006*"gun" + 0.006*"city" + 0.005*"man" + 0.005*"death" + 0.005*"family" + 0.005*"according" + 0.005*"shooting"'),
 (2,
  '0.009*"country" + 0.008*"government" + 0.007*"united" + 0.007*"military" + 0.006*"war" + 0.006*"attack" + 0.006*"force" + 0.006*"china" + 0.006*"syria" + 0.005*"north"'),
 (3,
  '0.008*"american" + 0.007*"think" + 0.006*"political" + 0.006*"obama" + 0.005*"dont" + 0.005*"white" + 0.005*"america" + 0.005*"way" + 0.005*"country" + 0.005*"going"'),
 (4,
  '0.006*"woman" + 0.005*"life" + 0.005*"show" + 0.004*"get" + 0.004*"know" + 0.004*"thing" + 0.004*"way" + 0.003*"story" + 0.003*"dont" + 0.003*"family"'),
 (5,
  '0.014*"game" + 0.011*"team" + 0.009*"player" + 0.005*"sport" + 0.005*"world" + 0.005*"season" + 0.005*"win" + 0.005*"photo" + 0.005*"league" + 0.004*"second"'),
 (6,
  '0.007*"health" + 0.006*"study" + 0.004*"drug" + 0.004*"use" + 0.004*"research" + 0.004*"data" + 0.004*"get" + 0.004*"company" + 0.003*"patient" + 0.003*"may"'),
 (7,
  '0.007*"city" + 0.006*"water" + 0.005*"climate" + 0.004*"car" + 0.004*"area" + 0.004*"home" + 0.003*"building" + 0.003*"space" + 0.003*"around" + 0.003*"change"'),
 (8,
  '0.013*"company" + 0.011*"percent" + 0.009*"million" + 0.008*"tax" + 0.007*"business" + 0.007*"market" + 0.007*"billion" + 0.006*"plan" + 0.005*"money" + 0.005*"price"'),
 (9,
  '0.009*"news" + 0.009*"house" + 0.007*"official" + 0.006*"email" + 0.006*"former" + 0.006*"investigation" + 0.006*"campaign" + 0.005*"russian" + 0.005*"white" + 0.005*"report"'),
 (10,
  '0.023*"republican" + 0.014*"vote" + 0.013*"campaign" + 0.013*"voter" + 0.012*"party" + 0.012*"election" + 0.011*"candidate" + 0.009*"democrat" + 0.009*"sander" + 0.008*"democratic"')]
```

We can also visualize this as follows. However, note that the topic indices labeled above and the indices labeled below in the diagram do not match.



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

- Topic 0: 8
- Topic 1: 6
- Topic 2: 4
- Topic 3: 2
- Topic 4: 1
- Topic 5: 11
- Topic 6: 7
- Topic 7: 10
- Topic 8: 3
- Topic 9: 5
- Topic 10: 9

In our analysis of the LDA model output, we identified overlaps between certain topics. Specifically, topics 4 and 5 (represented as 1 and 11 in the diagram) and topics 6 and 7 (represented as 7 and 10 in the diagram) exhibited considerable similarity in their thematic content. This similarity was evident in the shared keywords and underlying themes of these topics. Even though many keywords from topics 3 and 10 (represented as 2 and 9 in the diagram) overlap each other, we decided to separate these two topics because we did not want to simply narrow down them to “Politics”.

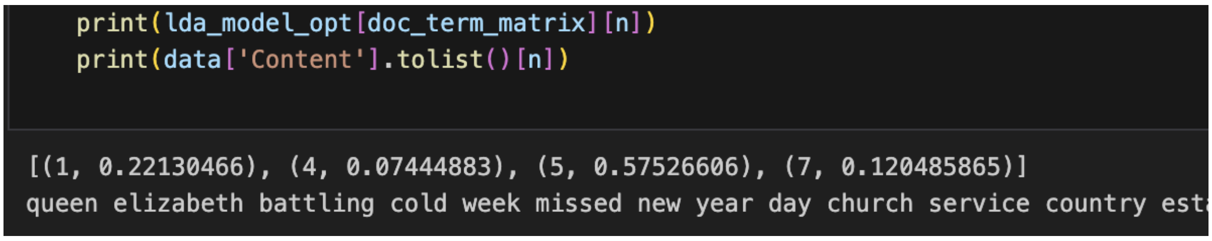
Furthermore, upon examining topic 4 (or 1 as illustrated in the diagram), we encountered challenges in delineating a distinct, coherent theme from its constituent words. This ambiguity further justified the need for consolidation.

As a result, we decided to merge topics 4 and 5 into a single topic, now referred to as topic 4. Similarly, topics 6 and 7 were combined into a new topic 6. This restructuring was based on the rationale that the themes within each pair of topics were closely related and their amalgamation would lead to more robust, clearly defined topics. The new grouping not only enhanced the clarity of the topics but also better represented the underlying patterns in the dataset.

Following is how we labeled each topic after observing their keywords:

- Topic 0: **Legal and Judicial Systems**
- Topic 1: **Court cases and Crime**
- Topic 2: **International Affairs and Military**
- Topic 3: **American Politics and Society**
- Topic 4 (4 and 5 combined): **Entertainment and Sports**
- Topic 6 (6 and 7 combined): **Health, Science, Environment, and Technology**
- Topic 8: **Business and Economy**
- Topic 9: **Politics and Media**
- Topic 10: **Political Campaigns and Election**

As you can see below, a document-term matrix that represents our corpus can be used to access the topic distribution of the n-th article. The output will be a list of tuples, where each tuple contains a topic identifier and the corresponding probability that this topic represents for the article.



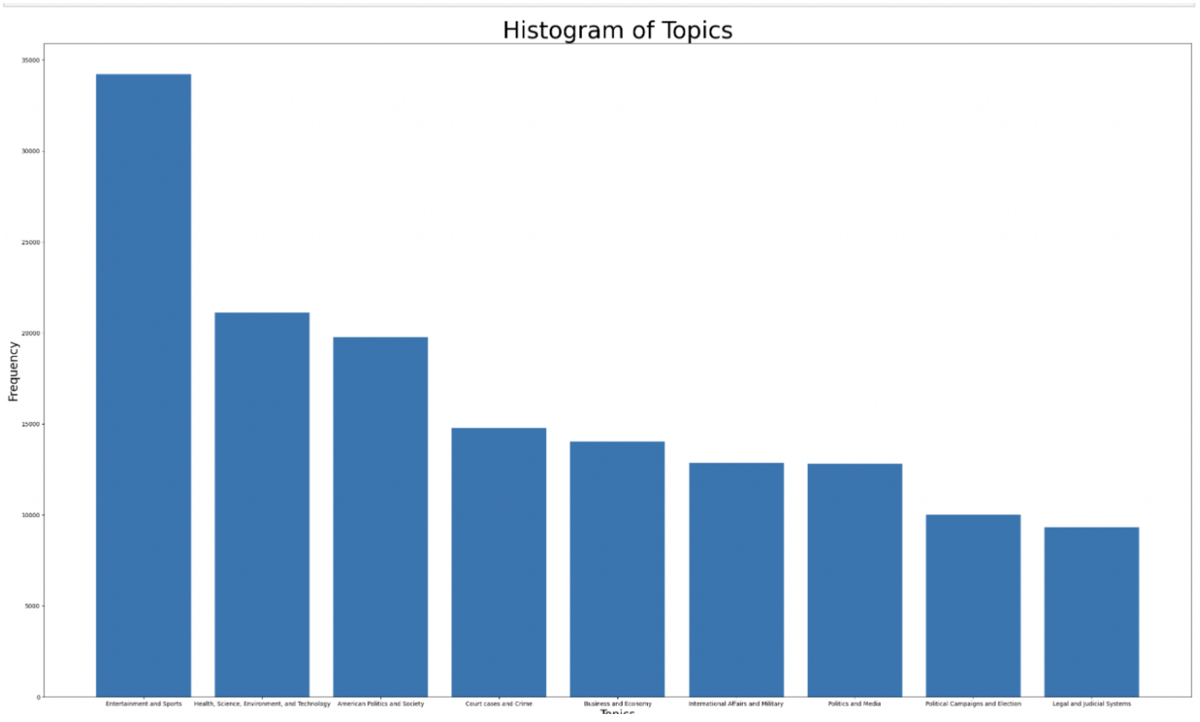
```
    print(lda_model_opt[doc_term_matrix][n])
    print(data['Content'].tolist()[n])
```

```
[(1, 0.22130466), (4, 0.07444883), (5, 0.57526606), (7, 0.120485865)]
queen elizabeth battling cold week missed new year day church service country est
```

Since we are interested in predicting exactly one label for each article, we decided to use the topic with the highest probability for its label. For the topics that we have combined (4, 5 and 6, 7), we summed up their probabilities.

Classification

In our project, we employed three distinct classification models: linear regression, LSTM, and DistilBERT. Our engagement with these three models was twofold, including both academic curriculum and supplementary external resources. The initial phase involved an analysis of our dataset by examining the distribution of news articles across the nine determined topics through a histogram. According to the analysis, the dataset revealed an imbalance, with the most number of news articles categorized as “Entertainment and Sports” and the least number of news articles categorized as “Legal and Judicial Systems.”

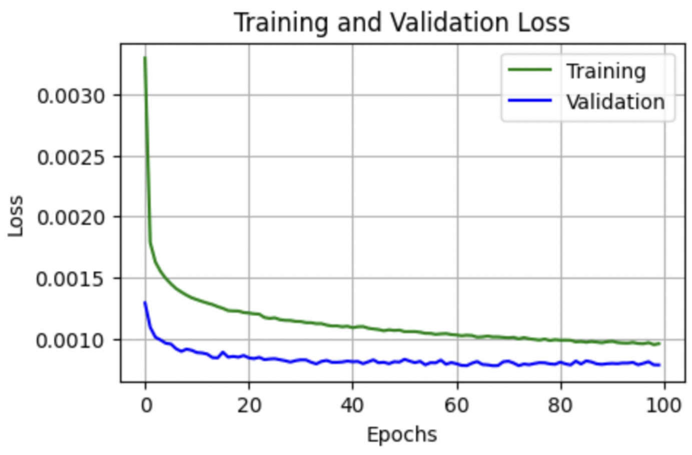


For the implementation of linear regression and LSTM models, we utilized 100d word embeddings from the Glove model. We began with constructing a three-layer linear regression classification model, a standard initial strategy in machine learning classification tasks. This is because it serves as a benchmark, enabling us to pursue more complex models based on its performance. All of the models utilized CrossEntropyLoss and Adam optimizer.

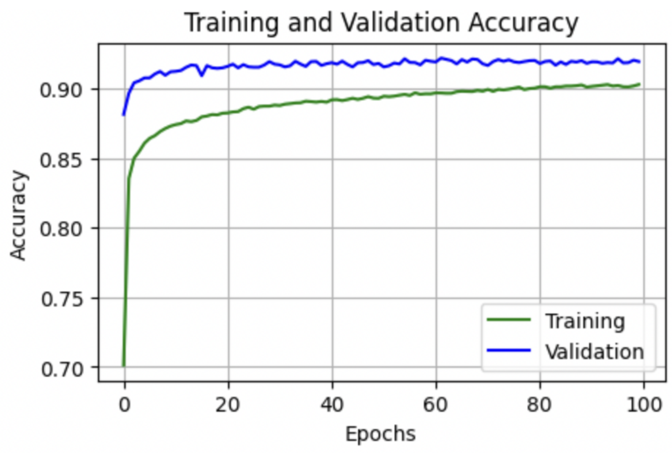
1. Linear

Our linear regression classification model mirrored the method employed in spam email detection, where we averaged the word embeddings from the Glove model if the word existed inside the model. We incorporated Rectified Linear Unit (ReLU) activation functions to introduce non-linearity and mitigate negative inputs, along with dropout layers to avoid overfitting. Furthermore, we used a batch size of 256, an epoch of 100, and a learning rate of 0.001.

After a series of experiments with varying hidden layer dimensions, we achieved a testing accuracy of 91.026% using a configuration of 256 and 64 as the hidden layer dimensions and a dropout rate of 0.3.



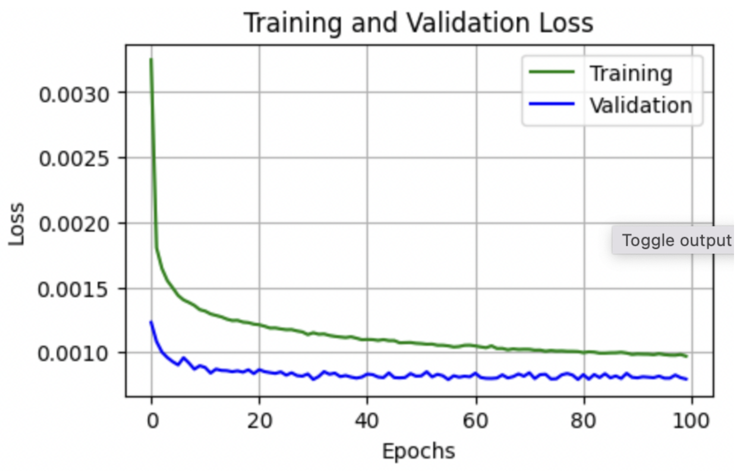
```
Final Training Loss:   0.000958
Final Validation Loss: 0.000784
```



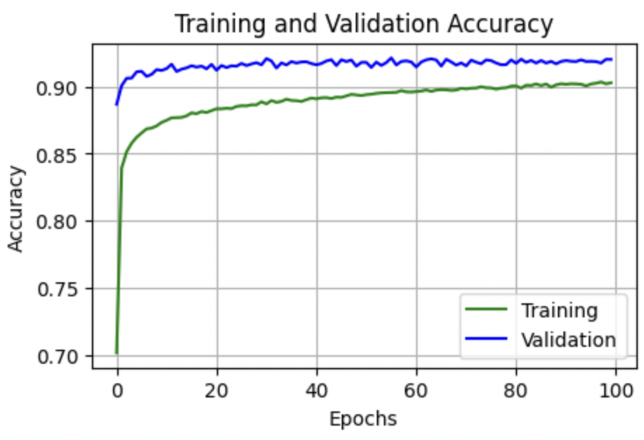
```
Final Training Accuracy:   0.903147
Final Validation Accuracy: 0.919661

Test Accuracy: 0.9102572798728943
```

We then refined the model with the addition of an extra layer, which slightly improved the accuracy to 91.120%. The modified model comprised 256, 64, and 256 hidden dimensions and maintained the dropout value of 0.3.



```
Final Training Loss:   0.00097
Final Validation Loss: 0.000794
```



```
Final Training Accuracy:   0.902802
Final Validation Accuracy: 0.920232

Test Accuracy: 0.9111977219581604
```

To evaluate our model's performance, we selected the following metrics from sklearn.metrics library:

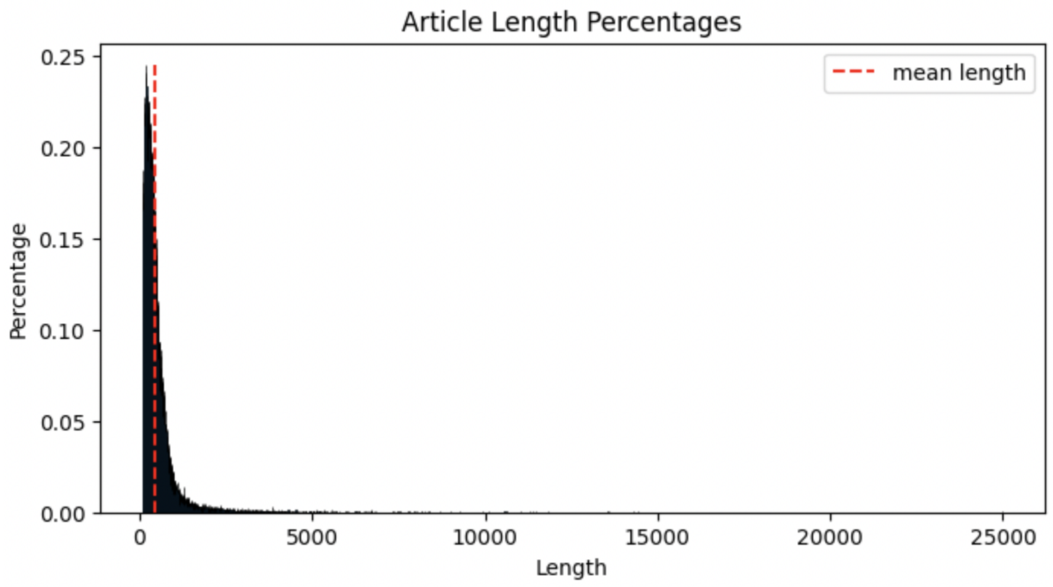
1. Accuracy score: This metric measures the proportion of correctly predicted instances to the total number of instances. It provides a general idea of the model's overall performance.
2. F1 score: The F1 score is the harmonic mean of precision (accuracy of positive predictions) and recall (proportion of true positives from the actual positives). Because the distribution of classes for our dataset is not uniform, it is particularly meaningful. It ensures that both false positives and false negatives are taken into account, providing a more balanced evaluation than accuracy alone.

The baseline accuracy of the news article dataset is 22.982%. Baseline precision, recall, and F1 scores were 2.55%, 11.11%, and 4.13%, respectively. The model outperformed the baseline accuracy by 68.138% and marked 0.8847 as the Cohen's Kappa score. The precision, recall, and F1 scores were 78.79%, 84.38%, and 73.74%, respectively, outperforming the baseline by 76.24%, 73.27%, and 69.61%.

Despite the high accuracy of 91.120%, we identified potential for further improvement by exploring other models. The linear regression classification model did not account for words outside of the Glove embeddings, which could have been critical in contextual understanding of the news articles. For instance, the word “Obama”, which was absent from the Glove embeddings, could indicate that the news article is related to politics.

2. LSTM

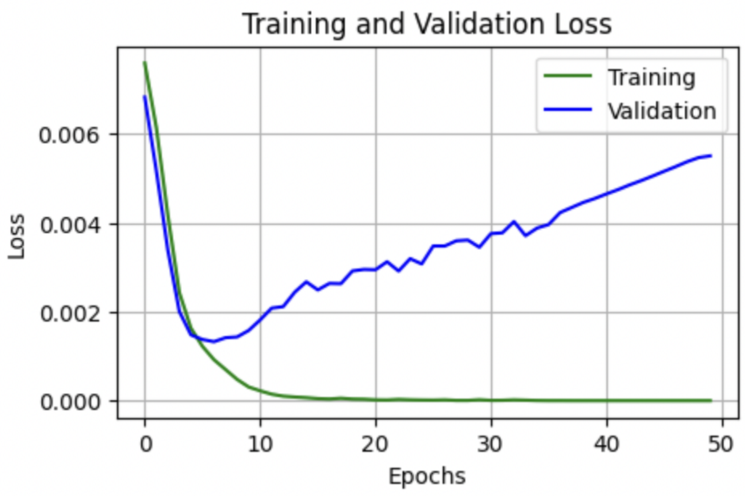
We implemented an LSTM model similar to the one used to parse the parts of speech in the Brown corpus by utilizing the Glove embeddings once again. For words excluded in the Glove dataset, we generated a random float between 0 and 1. We also filtered out articles under 100 words since we believed they would not provide meaningful results. Furthermore, we standardized the article lengths to the average number of words (440 words) to optimize training efficiency. Below is a detailed graph illustrating the distribution of words per news article within our dataset.



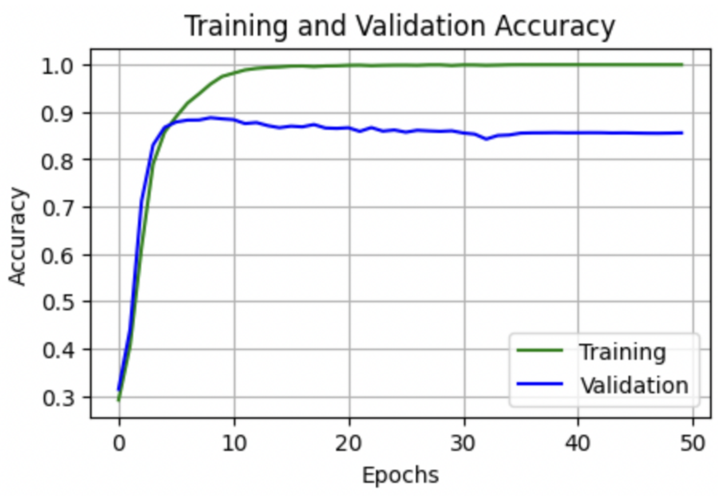
```
The average sentence length is 440.
```

This graph reveals that the majority of news articles consist of fewer words compared to the average number of words. It aligns with the intrinsic characteristics of news articles where pivotal keywords and critical arguments are predominantly positioned within the initial sections of the articles, allowing us to truncate later portions of the article that are less significant. Furthermore, the longest article in our dataset is composed of 24,974 words and we believe that extending every article by adding pad tokens is inefficient. Instead, our focus remained on grasping the essence of the content concisely and manageably.

The LSTM model was designed with two LSTM layers, incorporating dropout and linear regression. Similar to the linear model, we utilized a batch size of 256, a learning rate of 0.001, and an epoch of 100. Initial configurations with hidden dimensions of 256 and a dropout value of 0.3 resulted in overfitting the data, which was evident from the final accuracy of 1.0 and training loss of 0.0.



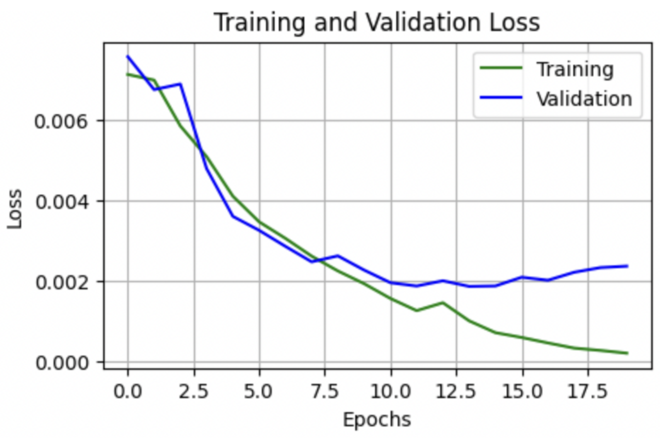
```
Final Training Loss:   0.0
Final Validation Loss: 0.005507
```



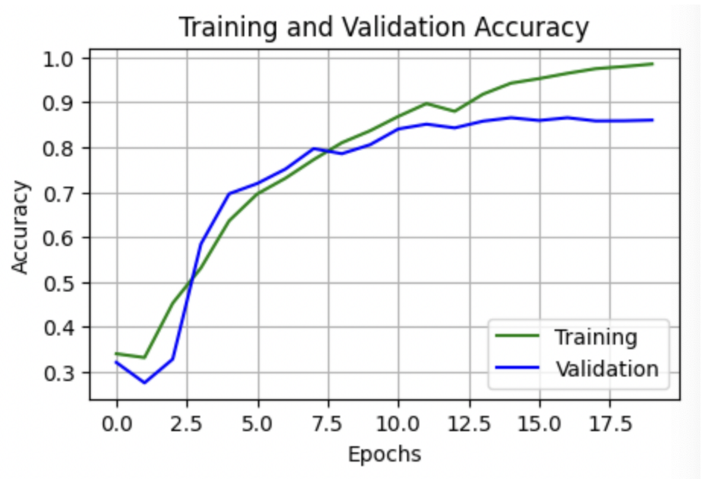
```
Final Training Accuracy:   1.0
Final Validation Accuracy: 0.855862

Test Accuracy: 0.836467981338501
```

Adjustments were made to reduce the hidden dimension to 64, leading to the following results. In addition, we recognized that the pattern of overfitting the data was evident between epochs 10 and 20, allowing us to reduce the epoch size to 20 for efficiency.



```
Final Training Loss:   0.000213
Final Validation Loss: 0.002375
```



```
Final Training Accuracy:   0.9847
Final Validation Accuracy: 0.859928

Test Accuracy: 0.8443105220794678
```

As shown above, the adjusted LSTM model achieved an accuracy of 84.431% without overfitting the data, surpassing the baseline accuracy but underperforming in comparison to the linear regression classification model. Such a result could be due to the standardization of the article lengths since it might have potentially omitted crucial information and the characteristics of LSTM models, which are typically more effective with sequential data. In other words, the categorization of topics may not greatly rely upon the sequential nature of the words in news articles, possibly explaining the lower accuracy of the LSTM model compared to the linear model.

By taking into account only the news articles that are longer than 100 words, the baseline accuracy is 23.049%. The baseline precision, recall, and F1 score were 2.57%, 11.11%, and 4.16%, respectively. Utilizing the same evaluation metrics as the linear model, the LSTM model indicated a 61.382% improvement over the baseline accuracy and a Cohen's Kappa score of 0.7977. The precision, recall, and F1 scores were 43.49%, 46.39%, and 43.77%, respectively, outperforming the baseline by 40.92%, 35.28%, and 39.61%.

3. Transformer

For our final model, we opted for a transformer-based model due to its advanced capabilities in processing natural language tasks. Transformer models like BERT and its variants such as RoBERTa and DistilBERT are at the forefront of NLP technology. We employed a distilbert-based-uncased model from Hugging Face's Transformers library, leveraging state-of-the-art capabilities in NLP tasks.

DistilBERT, a smaller, faster, and lighter version of BERT, retains most of its predecessor's performance capabilities. We thought DistilBERT was a perfect fit for our project since it is particularly suitable for environments where computational efficiency is a priority. Distilbert-base-uncased is a case-insensitive variant, implying that it does not differentiate between uppercase and lowercase letters. This is acceptable for our project because we run our model with a huge dataset, and we do not require case sensitivity for classifying articles.

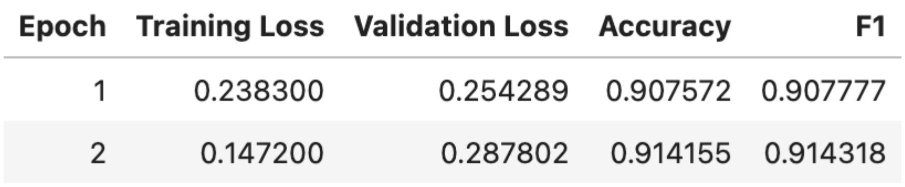
Tokenization is a crucial step in preparing text data, and it is important to choose the right tokenization method that fits the model. Therefore, we used the DistilBertTokenizer from Hugging Face, designed specifically for DistilBERT. It efficiently converts textual input into tokens that the model can comprehend, taking the model's vocabulary into account and handling special tokens required for DistilBERT's input. We decided to truncate or pad our articles into 512 tokens, the maximum number of tokens that the model can take.

Two prominent methods exist when using pre-trained models: training only the classification head and fine-tuning the model. Training only the classification head involves freezing the pre-trained model layers and only training the final dense classification layer. It is computationally less intensive but may not fully adapt the model to our specific task. Contrary to the first method, full model fine-tuning involves training the entire model on the new dataset. Although it is more resource-intensive, it allows the model to adjust all of its parameters to the new task, potentially leading to better performance. Hence, we chose to fine-tune the entire DistilBERT model, aiming to leverage its full capacity for adapting to our specific task of classifying news articles.

Below are the arguments used for training:

- Number of epochs: 2
- Learning rate: 2e-5
- Weight decay: 0.01
- Batch size: 16

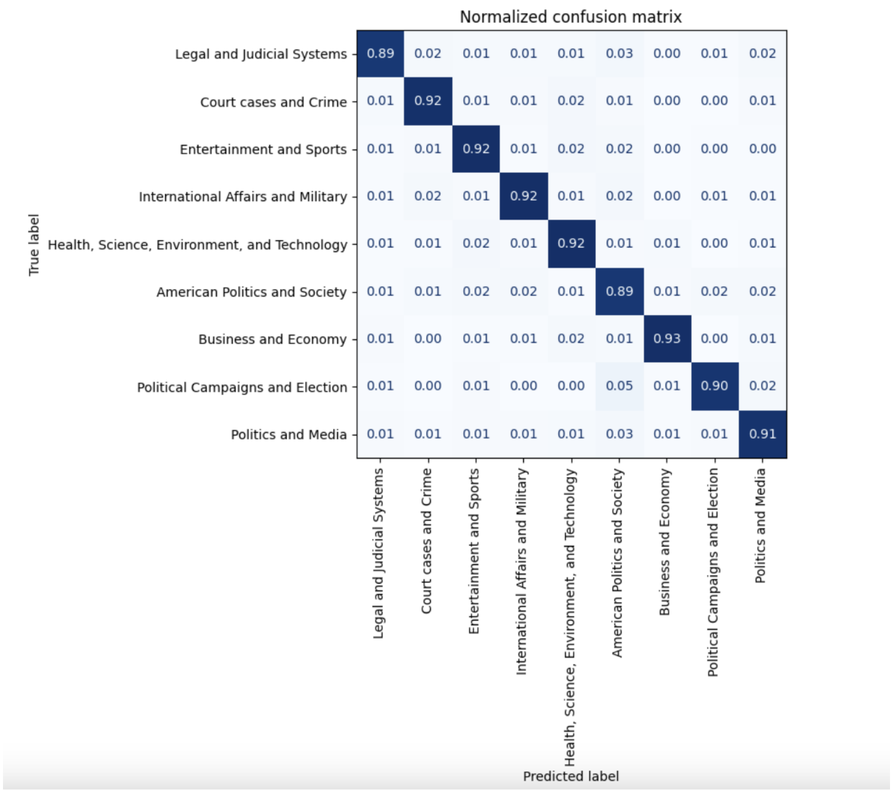
When fine-tuning pre-trained models, a small number of epochs is often sufficient to achieve high performance. Since our task is a simple text classification and these models have already learned general features in their pre-training phase, only a few epochs are required to adopt these features. The result shows that the fine-tuned model achieved 91.416% of accuracy, and 91.432% of F1 with only two epochs. Increasing the batch size can also help to increase the accuracy, but we could not run the model with a greater batch size due to the CUDA's memory problem. However, the transformer model outperformed the baseline accuracy and F1 by 68.434% and 87.282%, respectively.



Epoch	Training Loss	Validation Loss	Accuracy	F1
1	0.238300	0.254289	0.907572	0.907777
2	0.147200	0.287802	0.914155	0.914318

We also utilized an additional metric from sklearn.metrics library: confusion matrix. Given the non-uniform distribution of topics in our dataset, it is a crucial tool. It provides a detailed breakdown of the model's performance across different classes, showing where the model excels and misclassifies. This insight is vital for understanding the model's behavior in detail, especially in identifying biases or tendencies in misclassifying specific topics.

We can observe that the model often gets confused between “Political Campaigns and Election” and “American Politics and Society”, which is acceptable from a large number of intersecting keywords from these two topics observed during the topic modeling phase.



	Legal and Judicial Systems	Court cases and Crime	Entertainment and Sports	International Affairs and Military	Health, Science, Environment, and Technology	American Politics and Society	Business and Economy	Political Campaigns and Election	Politics and Media
Legal and Judicial Systems	0.89	0.02	0.01	0.01	0.01	0.03	0.00	0.01	0.02
Court cases and Crime	0.01	0.92	0.01	0.01	0.02	0.01	0.00	0.00	0.01
Entertainment and Sports	0.01	0.01	0.92	0.01	0.02	0.02	0.00	0.00	0.00
International Affairs and Military	0.01	0.02	0.01	0.92	0.01	0.02	0.00	0.01	0.01
Health, Science, Environment, and Technology	0.01	0.01	0.02	0.01	0.92	0.01	0.01	0.00	0.01
American Politics and Society	0.01	0.01	0.02	0.02	0.01	0.89	0.01	0.02	0.02
Business and Economy	0.01	0.00	0.01	0.01	0.02	0.01	0.93	0.00	0.01
Political Campaigns and Election	0.01	0.00	0.01	0.00	0.00	0.05	0.01	0.90	0.02
Politics and Media	0.01	0.01	0.01	0.01	0.01	0.03	0.01	0.01	0.91

Conclusion

This project successfully extended the principles of text classification to multi-class classification of news articles. By employing LDA, we categorized various news articles into 11 distinct topics suggested by the coherence score of 0.55. We agreed to merge some of the topics into one broader topic, concluding to categorize them into nine distinct topics. We constructed three different models: linear regression, LSTM, and DistilBERT. Among these models, the DistilBERT model showed the most remarkable accuracy of 91.416% and an F1 score of 91.43%.

If we were to continue to work on this area, we could explore multiple directions. Building further from our current project and dataset, we could dive deeper into various other models to further identify intricate patterns and relationships within the text, increase the batch size of the DistilBERT model, and experiment with a wide range of hyperparameters. Such an expansion could result in a more accurate classification. In addition, we can broaden the scope of our dataset by collecting news articles from a broader range of geographical locations and languages to improve its applicability. Finally, we can apply these classification methodologies to other domains, such as analyzing content on social media like Instagram or Facebook or in the realm of literary analysis.

Code

The links to the raw data (5 CSV files) are listed below.

- CNN articles: https://www.kaggle.com/datasets/hadasu92/cnn-articles-after-basic-cleaning
- BBC articles: https://www.kaggle.com/datasets/hgultekin/bbcnewsarchive
- Articles 1~3: https://www.kaggle.com/datasets/snapcrack/all-the-news

Along with this report, we submitted files that contain the code for cleaning data, topic modeling (LDA), and classification.

This GitHub link below also contains all of our code and links to the data:

https://github.com/jh000107/article_labeling

Contribution

As our initial plan, Jeong Yong collected data from five CSV files found on Kaggle. Following this, Junhui and Jeong Yong collaboratively worked to preprocess the text, ensuring only meaningful words were retained. After Junhui developed an LDA model and selected eleven as the most optimal number of topics, we merged some of them into one broader topic (topics 4,5 and 6,7) and identified labels for these nine topics. Jeong Yong worked on the linear regression and LSTM model while Junhui focused on the DistilBERT model. Concluding this project, we wrote the report together.