Predicting the Category of New York Times Articles Based on the Headline and the Keywords

Salman Faiz Hidayat[#1], Bambang Abhinawa Pinakasakti[*2]

[#]*Department of Computer Science, University of Gadjah Mada*
Bulaksumur, Caturtunggal, Depok, Sleman Regency, Special Region of Yogyakarta 55281
[1]salmanfaizhidayat@mail.ugm.ac.id
[3]bambangabhinawapinakasakti@mail.ugm.ac.id

[*]*OmahTI Division of Data Science of Artificial Intelligence*
69FG+QRP, Sagan, Caturtunggal, Depok, Sleman Regency, Special Region of Yogyakarta 55281

***Abstract*—** The problem is to accurately predict the category of an article, through its headline and keywords. The dataset comes from the New York Times. A Natural Language Processing (NLP) approach was used to make these predictions to solve this problem. Two models were used: an LSTM deep learning model and BERT Sequence Classifier model. The result of this experiment was that the BERT model performs better than the LSTM deep learning model, due to the fact that it is well pretrained. Other improvements that could be made include: increasing the number of samples, more sophisticated data preprocessing techniques, and a more simplified model.

***Keywords*—** Article Category, Long-Short Term Memory, BERT Sequence Classifier, Tokenizing, Sequencing.

I. Problem

This NLP (Natural Language Processing) analysis aims to solve the problem of classifying the category of an article clearly into its respective topic. In order to accomplish such task, a natural language data will be taken and processed, so that a model could accurately predict the category of a certain article.

II. Dataset

The dataset is from the Kaggle platform titled *New York Times Comments* by Aashita Kesarwani. For this project, the dataset was the article data from January-May 2017, and January-April 2018, with the articles data from April 2018 being used as the test dataset. The dataset contains the following columns:

1. Abstract: the abstract of the article that acts as the summary.
2. ArticleID: the ID of the article in the dataset.
3. articleWordCount: the number of words in the article.
4. byline: the column that contains information about the author.
5. documentType: the type of publication (article/blogpost)
6. headline: the title of the article.
7. keywords: important keywords about the article in the form of array words, that is itself converted to a string.
8. multimedia: (no description)
9. newDesk: the topic or the theme of the article.
10. printPage: the number of pages of the article.

It is important to note that the creator of this dataset does not provide any information about each of the columns. Descriptions provided here are inferred after processing it.

III. Methodology

In order to use NLP approach to predict the category article (column 'newDesk'), there are 2 variables that are taken into consideration :

1. headline
2. keywords

The selection is as such because these 2 columns in particular are the only ones that come in the form of string that mimics natural language, which allows for prediction using NLP. Here are example values for each variable with its headline-keywords value pair (format 'headline' : 'keywords'):

- 'Finding an Expansive View of a Forgotten People in Niger' : '['Photography', 'New York Times', 'Niger', 'Ferguson, Adam (1978-)', 'Boko Haram']'
- 'And Now, the Dreaded Trump Curse' : '['United States Politics and Government', 'Trump, Donald J', 'Spicer, Sean M (1971-)', 'Ryan, Paul …'
- 'Venezuela's Descent Into Dictatorship' : '['Venezuela', 'Politics and Government', 'Maduro, Nicolas']
- 'Stain Permeates Basketball Blue Blood' : '['Basketball (College)', 'University of North Carolina', 'Williams, Roy A', 'Cheating', 'NCAA Basket…'
- etc.

After obtaining the dataset and determining the variables necessary for the task

A. Data Preprocessing

Data preprocessing are divided into 4 steps:

1. Data preparation.
2. Binning and encoding target labels.
3. Text cleaning.
4. Concatenating 'headline' and 'keywords' column

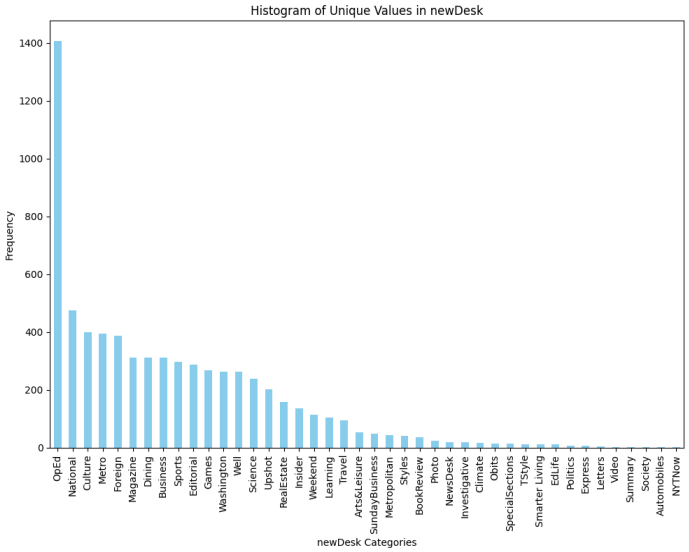
Data preparation starts by concatenating the articles within January - May 2017, and those from January - March 2018. Then, the article data from April 2018 are left untouched, which will be used as a separate test dataset. Afterwards, columns other than 'headline' and 'keywords' are removed from the dataset, in order to narrow down the focus of the analysis to only natural language.

Furthermore, the dataset contains values that are unknown or empty. For 'headline,' some empty data was labeled as 'Unknown' and some keywords were labeled as '[],' indicating that there is no keyword. So in order to treat those, those labels are replaced with 'NaN,' so that it can later be dropped from the dataset. The reason for why dropping values were chosen was because empty values will confuse the model from actually predicting, and it is impractical to impute missing values, and almost impossible if they are in the form string data type.

Next, the data preparation is followed by **binning the targets**. The target is the 'newDesk' column. Here are the unique values of the said column with its respective histogram for it:

- 'National', 'Foreign', 'Games', 'OpEd', 'Metro', 'Sports', 'Culture', 'Editorial', 'Science', 'Upshot', 'Magazine', 'Business', 'SundayBusiness', 'RealEstate', 'Well', 'Dining', 'Styles', 'Insider', 'Learning', 'Weekend',

'BookReview', 'Travel', 'Metropolitan', 'Politics', 'EdLife', 'Letters', 'Arts&Leisure', 'SpecialSections', 'Investigative', 'NewsDesk', 'Summary', 'Automobiles', 'Society', 'Washington', 'Smarter Living', 'Climate', 'Obits', 'Express', 'Video', 'TStyle', 'Photo', 'NYTNow'

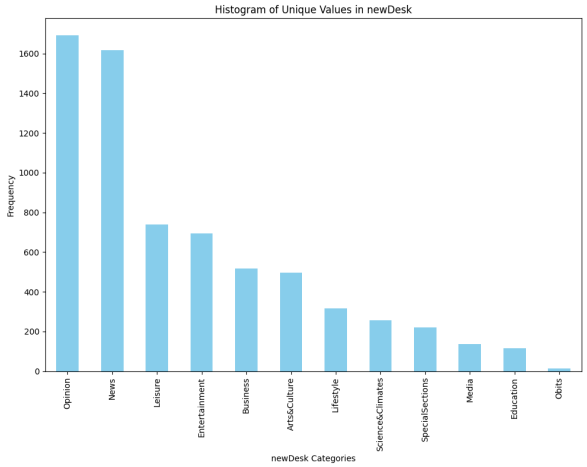


By observing the histogram, it is apparent that OpEd is the most frequent article category in the dataset, followed by national and culture. For context, OpEd is an article that contains a writer's strong and focused opinion about a certain matter [1].

To accommodate for the large cardinality, the categories are binned together. The reason for this is to minimize noise as much as possible, so that the model can properly learn [2]. The following is the binning scheme that made (format 'binned_label' : [actual_labels]) :

- 'News' : ['National', 'Foreign', 'Politics', 'Washington', 'Investigative', 'Metro', 'NewsDesk', 'Express', 'NYTNow', 'Metropolitan']
- 'Entertainment' : ['Sports', 'Games', 'Travel', 'BookReview']
- 'Arts&Culture' : ['Culture', 'Arts&Leisure', 'TStyle', 'Letters', 'Photo']
- 'Business' : ['Business', 'SundayBusiness', 'RealEstate', 'Automobiles']
- 'Science&Climates' : ['Science', 'Climate']
- 'Lifestyle' : ['Smarter Living', 'Well', 'Styles', 'Society']
- 'Opinion' : ['OpEd', 'Editorial']
- 'Leisure' : ['Dining', 'Magazine', 'Weekend']
- 'SpecialSections' : ['SpecialSections', 'Summary', 'Upshot']
- 'Education' : ['Learning', 'EdLife']
- 'Media' : ['Video', 'Photo', 'Insider']
- 'Obits' : ['Obits']

After binning, the target column will have less unique values. The following is the histogram of the target column after binning:



It can be observed that there are significantly less categories for the model to predict, which will supposedly improve performance.

The next preprocessing step is **encoding the target labels.** To label encode two dictionaries were made, namely id2label and label2id. The label2id is used for mapping the label into integers on the dataframe by assigning the label as the key and a number as the value. While, id2label is used for translating the encoded labels back to its original form by assigning the number as the key and the label. Below are the encoding of the said dictionaries:

- label2id: {

  'news': 0, 'entertainment': 1, 'opinion': 2, 'arts&culture': 3, 'science&climates': 4, 'specialsections': 5, 'leisure': 6, 'business': 7, 'lifestyle': 8, 'media': 9, 'education': 10, 'obits': 11

  }
- id2label : {

  0: 'news', 1: 'entertainment', 2: 'opinion', 3: 'arts&culture', 4: 'science&climates', 5: 'specialsections', 6: 'leisure', 7: 'business', 8: 'lifestyle', 9: 'media', 10: 'education', 11: 'obits'

  }

The final steps of data preprocessing is **text cleaning and the concatenation** of cleaned 'headline' and 'keywords' text. Stopwords are common words that do not necessarily contribute any meaning, such as 'the,' 'an,' 'a,' etc. By removing them, it will increase the efficiency of the model by drawing its attention towards more important words [3], and the same applies for punctuations [4]. For detecting stopwords, a stopwords library from NLTK was used. With all of that in consideration, a function was defined to clean the text. Below is the logic of the clean_text() function.

- Turning every word to lowercase.
- Removing URLs, HTML tags, and punctuations.
- Removing stopwords using NLTK library.
- Removing special characters and extra spaces.
- Lemmatizing each of the cleaned words.

Additionally, lemmatizing is the process of grouping similar with different forms, into one single form. This is done so that the model could identify the words when they are in their base form. Furthermore, it reduces data size, and helps with search results, since a single base form of a word can be matched with many words. One simple example would be lemmatizing the word "walked" to its base form "walk" [5].

After cleaning the text in both 'headline' and 'keywords' column, the cleaned texts are concatenated together into one single column. This is done so that the input shape for the model will be simpler, namely one input column and one target column. Finally,

the cleaned data is split into train, validation, and test.

B. Data Analysis

Data analysis is done after preprocessing so that the correct parameters for the model can be determined, namely the num_words parameter for the tokenizer, and maxlen parameter for sequencing of the words.

Tokenization is the process of encoding words into numbers. In the code implementation, tokenizer is an object that can tokenize the language data, into smaller units of encoded tokens. One parameter in particular, namely 'num_words,' determines how many words to be tokenized [6].

```
sentences = [
    'I love my dog',
    'I love my cat'
]
```

```
{'i': 1, 'my': 3, 'dog': 4, 'cat': 5, 'love': 2}
```

For words that are not included in the 'num_words,' an OOV token (out-of-vocabulary) is used. This way, it preserves the length of the sentence, while still taking them into account without disregarding them in the sequence [7].

```
sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
]
```

```
[[5, 1, 3, 2, 4], [2, 4, 1, 2, 1]]

{'think': 9, 'amazing': 11, 'dog': 4, 'do': 8, 'i': 5, 'cat': 7,
 'you': 6, 'love': 3, '<OOV>': 1, 'my': 2, 'is': 10}
```

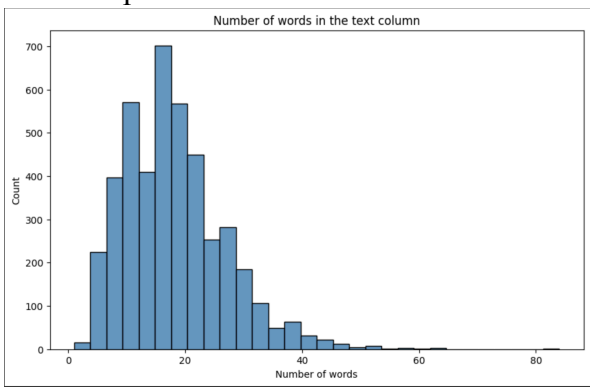
Sequencing is a process of turning tokens into a sequence of tokens that represents sentences [7]. The sequence consists of arrays where each array represents a sentence with tokenized words. One important parameter in this step is the 'maxlen' parameter, which determines the maximum length of a sentence in the sequence of tokenized sentences.

```
{'do': 8, 'you': 6, 'love': 3, 'i': 5, 'amazing': 11, 'my': 2,
'is': 10, 'think': 9, 'dog': 4, '<OOV>': 1, 'cat': 7}

[[5, 3, 2, 4], [5, 3, 2, 7], [6, 3, 2, 4], [8, 6, 9, 2, 4, 10, 11]]

[[ 0  0  0  5  3  2  4]
 [ 0  0  0  5  3  2  7]
 [ 0  0  0  6  3  2  4]
 [ 8  6  9  2  4 10 11]]
```

The main interest of this analysis is the number of unique words that make up most of the dataset. In this case, words that are taken into account are words that make 95% percent of the vocabulary in the dataset. It was found that there were 13612 unique words, and there are 7493 unique words that make up 95% of the dataset. The reason for not including every vocabulary in the dataset is to increase efficiency, by focusing only on the most relevant words, and discards less frequent words that do not contribute to the performance [8]. So therefore, 'num_words' parameter is set to 7493. The tokenizer will tokenize all of these, and then use OOV tokens for the rest.

Next, the sentence length was analyzed to determine the best value for 'maxlen' parameter. After analyzing the length of sentences in the dataset, it was found that most sentences were around 40 words long, where most of the sentences populate the range of 5 - 60 words. So therefore, the

‘maxlen’ parameter will be set to 60.



C. Modeling

There are 2 models was used as experiment for this task:

1. LSTM (Long-Short Term Memory) Deep Learning Model [10][11][12], without binning the target.
2. BERT for Sequence Classification [9], with target binning .

Here is the architecture of the first model (in order):

1. Conv1D (filters=128, kernel_size=5, activation='relu')
2. MaxPooling1D (pool_size=2),
3. Conv1D (filters=128, kernel_size=5, activation='relu')
4. MaxPooling1D (pool_size=2),
5. LSTM (layers.LSTM(128, return_sequences=True)),
6. LSTM (layers.LSTM(128)),
7. BatchNormalization
8. Dense (512, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)),
9. Dropout (0.5)
10. BatchNormalization
11. Dense (256, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.01)),
12. Dropout (0.5)
13. Dense (softmax)

It is important to note that only the first model requires the analysis hitherto mentioned. This is the case because LSTM models require the data to be in the form of a tensor, and be tokenized by TensorFlow’s tokenizer; this tokenizer requires the ‘maxlen’ and ‘num_words’ parameters.

In addition, it is important to recognize the role of each layer. Here is what each layer does:

1. Convolutional 1 Dimension: This layer is used to capture any pattern in the sentence [13][14].
2. LSTM: This layer is mainly used to capture the overall context of a sentence, due to its ability to keep important information by the virtue of its Cell State, which can bring context to every timesteps [11].
3. Dropout: dropout layers are supposed to block a fraction of neurons randomly. The fraction is specified in the parameter [15].
4. Dense layer with ReLU activation: these layers are provided to capture any important information that might have been missed by the previous layers.
5. Dense with Softmax activation: this is the output layer that predicts multiple article categories.

   “The softmax activation function transforms the raw outputs of the neural network into a vector of probabilities, essentially a probability distribution over the input classes. Consider a multiclass classification problem with N classes. The softmax activation returns an output vector that is N entries long, with the entry at index i corresponding to the probability of a particular input belonging to the class i.” [16]

Secondly, the BERT for Sequence model is used. In this case, the previous analysis is not used because this model is pre-trained and employs a transformer-based architecture, which fundamentally differs from LSTM-based models.

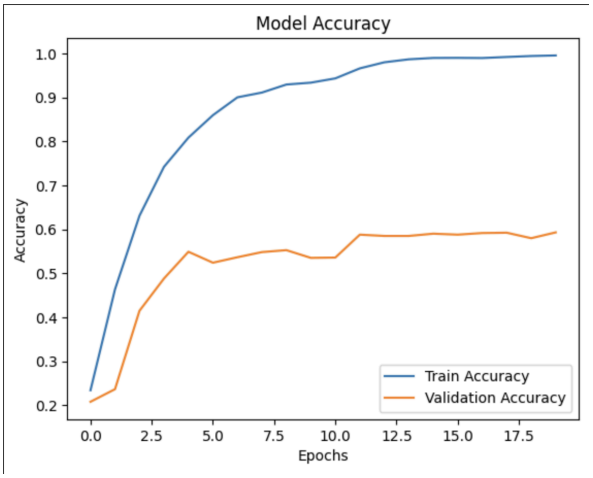
Instead, BERT leverages self-attention mechanisms that focus on all words in a sentence simultaneously, creating rich contextual embeddings. This allows BERT to understand long-term dependencies and relationships between words better.

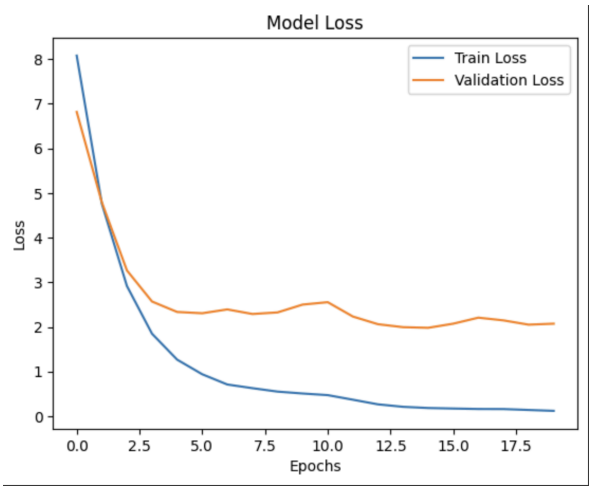
Below is the architecture of the BERT for Sequence Classification model:

1. BERT Encoder: The input is tokenized and passed through BERT's transformer layers. It contains sub-layers of multi-head attention and feedforward networks that work together to encode the text.
2. Classifier Head: After BERT Encoder layers, the token that is used to aggregate information from the entire sequence (CLS) is passed to a dense classification layer where the final layer uses softmax activation to predict the target categories.

IV. Review of Results

For the first model, the model had an average F1 score of 0.5, with a validation loss of around 2. In the test section, the model could only get an F1 score of 0.1. By assessing the loss and accuracy curve, the model also seems to overfit:





The second model had a weighted average F1 score of 0.71 on the split test data and 0.73 on unseen data. It performs well on categories like news, entertainment, and opinion. Specifically, entertainment achieved the highest F1-score of 0.88 on split test data and 0.87 on unseen data. On the other hand, categories like specialsections and science&climates had lower F1-scores. Furthermore, the obits category had an F1-score of 0. This is because there are very few rows that have this label.

Here is the table of classification report for split test data

	Precision	Recall	F1-Score
News	0.80	0.80	0.80
Entertainment	0.83	0.93	0.88
Opinion	0.80	0.68	0.74
Arts&Culture	0.63	0.86	0.72
Science&Climates	0.48	0.43	0.45
SpecialSections	0.34	0.44	0.38
Leisure	0.65	0.57	0.61
Business	0.68	0.65	0.67
Lifestyle	0.66	0.70	0.68
Media	0.35	0.41	0.38
Education	0.61	0.74	0.67
Obits	0.00	0.00	0.00
Accuracy			0.71
Macro Average	0.57	0.60	0.58
Weighted Average	0.72	0.71	0.71

Here is the table of classification report for unseen data

	Precision	Recall	F1-Score
News	0.80	0.79	0.80
Entertainment	0.86	0.87	0.87
Opinion	0.76	0.73	0.74
Arts&Culture	0.75	0.89	0.81
Science&Climates	0.50	0.51	0.51
SpecialSections	0.75	0.74	0.38
Leisure	0.51	0.55	0.53
Business	0.75	0.74	0.75
Lifestyle	0.73	0.49	0.59
Media	0.31	0.50	0.38
Education	0.33	1.00	0.50
Obits	0.00	0.00	0.00
Accuracy			0.73
Macro Average	0.55	0.63	0.57
Weighted Average	0.73	0.73	0.71

V. Conclusion

In conclusion, the BERT for Sequence Classification model performs better at predicting an article category than the traditional TensorFlow deep learning models, mainly due to the fact that BERT is pretrained. Other improvements that could be made include:

- Larger dataset for the data to train on, because the current only has around 8000 samples, and around 5000 after preprocessing. Larger dataset is necessary because natural language or text is a very unstructured data.

- A more advanced techniques of preprocessing natural language data could also help improve the performance of the model.
- In addition to advanced preprocessing techniques, it should also be paired with a model that is more efficient with fewer neurons, so that it is less susceptible to overfitting. This is especially true for the first model.

References

[1] "Definition of op-ed". Merriam-Webster Online Dictionary. Retrieved June 29, 2010.

[2] https://towardsdatascience.com/feature-engineering-examples-binning-categorical-features-9f8d582455da

[3] https://www.geeksforgeeks.org/removing-stop-words-nltk-python/

[4] https://www.kaggle.com/code/abdmental01/text-preprocessing-nlp-steps-to-process-text

[5] https://www.geeksforgeeks.org/python-lemmatization-with-nltk/

[6] https://www.youtube.com/watch?v=fNxaJsNG3-s&list=PLBLDLru7DwE7BVbwsUdCoxtHHP-PnKwbo

[7] https://www.youtube.com/watch?v=r9QjkdSJZ2g

[8] https://eitca.org/artificial-intelligence/eitc-ai-tff-tensorflow-fundamentals/natural-language-processing-with-tensorflow/tokenization/what-is-the-tensorflow-keras-tokenizer-api-maximum-number-of-words-parameter/#:~:text=In%20practice%2C%20setting%20the%20%60num_words,significantly%20to%20the%20model's%20performance.

[9] https://github.com/rohan-paul/LLM-FineTuning-Large-Language-Models/blob/main/Other-Language_Models_BERT_related/FineTuning_BERT_for_Multi_Class_Classification_Turkish/Multi-class_Classification.ipynb

[10] https://www.youtube.com/watch?v=OuYtk9Ymut4

[11] https://www.youtube.com/watch?v=A9QVYOBjZdY

[12] https://www.youtube.com/watch?v=ZMudJXhsUpY

[13] https://www.superannotate.com/blog/guide-to-convolutional-neural-networks#the-convolutional-layer

[14] https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns

[15] https://www.kaggle.com/code/ryanholbrook/dropout-and-batch-normalization

[16] https://www.pinecone.io/learn/softmax-activation/