Breaking News! Which Category Is It?

Group 20: Oon Zhi Xiang, Branson Ng Khin Swee, Wen Xin, Fang Pin Sern[a)]

(Dated: April 23, 2021)

I. Introduction

News is published at a very fast pace these days. With these large amounts of news published daily, it is important we distinguish the news categories from the headlines itself so we could read the news that we are most interested about. Using Natural Language Processing Techniques, we hope to generate classification labels for the news accurately.

The dataset we are working on is the News Category Dataset [3]. We extract headlines of 10 categories (CRIME, ENTERTAINMENT, FOOD DRINK, MONEY, RELIGION, SPORTS, STYLE, TECH, TRAVEL and WOMEN) and try to classify the headlines into their corresponding categories.

II. Related Work

In general, a text classification technique can be categorized as statistical or belonging to machine learning [4]. Statistical approaches together with dimension reduction techniques like Principle Component Analysis [2] tend to work well with linear data. However, as the dataset becomes large and non-linear, machine learning approaches become more preferred. For this project, we are interested in the supervised learning approaches.

The supervised learning approaches can be divided into parametric and non-parametric approaches [4]. Parametric models like logistic regression, naive bayes, perceptron and simple neural network etc. have a limited number of parameters and do not scale with the increase in sample size. They tend to be speedy in training but in general give poor fit to the data [1]. In contrast, non-parametric models like k-nearest neighbours and support vector machine (SVM) would scale with the sample size. A huge and complex neural network, albeit its parametric nature, would be able to emulate the effect of a non-parametric model with its huge number of parameters when the sample size gets huge. The non-parametric approaches tend to be slow in training but in general give good fit or even over-fit to the data [1]. In our project, we would use a parametric model as the baseline model, and try to obtain better classification performances using larger scaled neural networks.

III. Methods

A. Understanding the Corpus and Pre-processing Techniques

In this section, we will introduce the corpus we worked on and the pre-processing methods used. We will firstly take a look at the dataset to gain a rough idea of how the corpus looks like. Then, we will talk about the intuitions and details for the pre-processing methods. The justifications for their performances will be provided in Section V A 2.

1. Understanding the Corpus

As shown in Figure 1, the dataset is imbalanced, with ENTERTAINMENT significantly outnumbering other categories. Figure 2 shows that the headlines are in general short, with almost all headlines having 20 words or below in terms of length. The longest headline belongs to ENTERTAINMENT: *Chats with Esperanza Spalding, Michelle Phillips, Lee Greenwood, Ian Thomas and Young Gun Silver Fox's Shawn Lee, Plus Joey Alexander, Elayna, Ultan Conlon, M Ross Perkins, Morgan's Road, Deerheart, Dave McGraw Mandy Fer, Unconscious Disturbance, I The Mighty, and The Junior League Exclusives*, and the shortest headline is a hashtag: *#WhyWeMarched*.

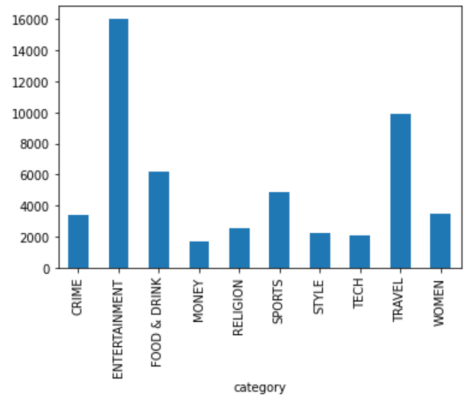

Figure 1: Distribution of headlines in terms of category. Horizontal axis: category. Vertical axis: the number of headlines in the category.

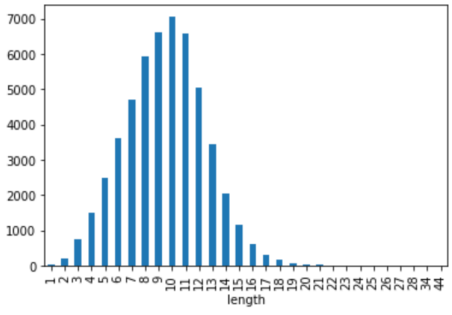

Figure 2: Distribution of headlines in terms of the number of words of the headline (length). Horizontal axis: length of the headline. Vertical axis: the number of headlines of the length.

2. General Text Pre-processing Techniques

We have explored 5 general text pre-processing techniques, including 4 tokenization methods and balancing the dataset. The intuitions and details of these

[a)]Electronic mail: e0310291, e0309295, e0052753, e0321477@u.nus.edu; Code Repository: https://github.com/fangpinsern/CS4248-Project

general techniques are introduced below. In addition to these 5 general techniques, there are some pre-processing techniques specific to the models used, and they will be introduced together with their models respectively. The performances of the text-preprocessing techniques applied on the models will be introduced in Section IV and justified in Section V.

tokenize: we noticed that there are in total 217 headlines that contain hashtags in the dataset, and these hashtags can contain information critical for categorization. This tokenization method will break the text into unigrams. In the process, hashtags are broken down into unigrams and stopwords are retained. Example: *'#TrumpHair Is The Hilarious Hashtag Our Country Deserves'* → [*'trump'*, *'hair'*, *'is'*, *'the'*, *'hilarious'*, *'hashtag'*, *'our'*, *'country'*, *'deserves'*].

tokenize_bigram: we noticed the frequent appearances of fixed 2-word or 3-word expressions. These bigrams and trigrams are also indicative of the categories. For example, the best appears in 5% the FOOD DRINK headlines and 3% of the STYLE headlines, but does not appear at all in CRIME, RELIGION, TECH, SPORTS, and WOMEN. Hence we want to incorporate frequently appearing bigrams into our categorization. This tokenization method is built upon **tokenize**, and will in addition group the words that frequently appear together into bigrams. Example: *'Kylie Jenner Launches #IAmMoreThan Anti-Bullying Campaign On Instagram'* → [*'kylie_jenner'*, *'launches'*, *'i_am'*, *'more_than'*, *'anti'*, *'bullying'*, *'campaign'*, *'on'*, *'instagram'*].

tokenize_synonym: we noticed that many headlines in the same category shared keywords of similar meanings. Hence, we assume that words in the same category tend to be more similar in meaning than words in different categories. In other words, the words in the same category can be represented using some synsets, and it is unlikely that other categories share the same synsets. This tokenization method is built upon **tokenize**, and will in addition substitute each word token into another word token which represents the synset cluster of the original word token. Example: *'Eating Healthy Just Got Easy With These 20 Recipes (PHOTOS)'* → [*'consume'*, *'goodly'*, *'barely'*, *'arrive'*, *'comfortable'*, *'twenty'*, *'recipe'*, *'photograph'*].

tokenize_hypernym: the intuition for using hypernyms is similar to that of using synonyms. This tokenization method is built upon **tokenize**, and will in addition substitute each word token into another word token which represents the hypernym cluster of the original word token. Example: *'Eating Healthy Just Got Easy With These 20 Recipes (PHOTOS)'* → [*'consume'*, *'annoy'*, *'large'*, *'integer'*, *'direction'*, *'representation'*].

balancing dataset: given that the dataset is imbalanced in terms of category, we want to see if balancing the dataset would improve the classification performances. This technique would upsample or downsample the headlines in each category to make sure that the numbers of samples in each category are equal.

B. Logistic Regression

Under the logistic regression model, we look to explore the effectiveness of its classification and use it as a baseline comparison to the result of other models.

Both TFIDF Vectorizers and Count Vectorizers were used to extract features from the data. This is done to compare which gives a better feature set for the linear regression to work with.

Name Tagging is a preprocessing technique applied here given the importance of names in distinguishing the categories. For example if the headline is "Mother Of Pittsburgh Pirates Catcher Elias Diaz Kidnapped In Venezuela" will become "Mother Of ORGANIZATION ORGANIZATION Catcher PERSON PERSON Kidnapped In LOCATION"

C. Feed-forward Neural Network

1. Feature Vectors

For this model, we used Glove vectors as the feature vectors for the word tokens. To use a single vector to represent a headline, we did the following: for a headline $h = [X_1, X_2, ...X_n]$, $h \in R^{d \times n}$ where $X_i \in R^d$ is the feature vector representing the i^{th} token in the headline. The feature vector representing the headline h would be obtained by

$$v(h) = \frac{\sum_{i=1}^{n} h_i^T}{\sqrt{\sum_{j=1}^{d} \left(S(h_j)\right)^2}}$$

where h_i is the i^{th} row of h, $S(u) = \sum_{i=1}^{m} u_m$ for $u \in R^m$.

In our experiments, we tried both 50d and 300d Glove vectors and found that 300d Glove vectors produced a better macro F1 score. The results of applying different text pre-processing techniques on feed-forward neural network shown in Table I and II use 300d Glove vectors as the feature vectors.

2. Training

We used a neural network with the following architecture: 1 input layer of 300 nodes, 2 hidden layers of 300 nodes with Dropout Regularization of 0.4 and 1 output layer of 10 nodes. Early Stopping was also used with a patience of 3 epochs. We trained for a total of 100 epochs, but due to Early Stopping, training usually stops at around the 10 epoch mark.

D. RNNs and Transformers

In this section we will be talking about two main architectures, RNN and transformers. In particular we used a LSTM and a DistilBert model. Subsections prepended with **Approach** are variants we have experimented with on the DistilBert model and can be found in Table I.

1. Data

For both models, a weighted sampler was used during the training phase to alleviate the imbalance in the dataset. This allows us to train with a more uniform category distribution and should improve the F1-score.

Approach - Synonym Augmentation

Augmentation of the data was also done in the training phase to reduce overfitting as can be seen in Section

VII.A. We augment the data by replacing words with a synonym chosen randomly via WordNet's synset. Using POS tags to ensure that the synonym was of the same class as the original word improved the quality of the augmentation. **Example:** *Original Text:* "Which Celebrities Did Not Bite Beyoncé's Face? An Updating List." *Augmented Text:* "'which famous person did not bite beyoncé's human face an updating listing". It differs slightly from the previous tokenize_synonym method by randomly choosing to use the synonym 30% of the time instead of 100% of the time. This approach is expected to better retain the semantics of the sentence. The value 30% was chosen empirically.

2. Embeddings

Both the LSTM and DistilBert models used pretrained embeddings. LSTM uses the Glove.6B 50 dimensions embeddings and DistilBert uses the pretrained embeddings layer from "distilbert-base-uncased". The embedding layers were then fine-tuned during training.

Approach - Embeddings for Unknown Tokens

Unknown tokens are somewhat common in some categories like the ENTERTAINMENT categories. As such, we decided to try expanding the vocabulary of our embeddings to include unknown tokens found in the training data. The vectors for the new vocabulary were initialized from a normal distribution. We then trained the model embeddings.

3. Training

Dropout layers and weight decay were used to regularize the model and discourage overfitting.

Hyperparameters: weight decay of 0.2, learning rate of 2e-4, ADAM optimizer and a scheduler that steps the learning rate by a factor of 0.2 every 2 epochs. We trained the DistilBert models for 5 epochs as we only needed to fine-tune the pretrained models. Larger batch size of 512 was used for the transformer as it helped reduce the gradient noise scale during optimization. Since there were no pretrained weights for the LSTM, we had to train the model for 20 epochs to achieve some level of accuracy.

4. LSTM

A simple single layer, unidirectional LSTM was used with a hidden and cell state size of 128 dimensions.

5. LSTM + Attention

The Attention layer contains a fully connected linear layer that generates an attention score by calculating a dot product of the final hidden state with hidden states at each time step of the input. We then get a weighted average of all hidden states by multiplying the attention to the hidden states at each time step.

6. DistilBert Transformer

We used the pre-trained DistilBert transformer from HuggingFace. DistilBert was chosen as it has has fewer parameters, runs 60% faster while preserving over 95% of BERT's performances as measured on the GLUE language understanding benchmark. The "distilbert-base-uncased" variant was used since we do lower the case of our inputs when preprocessing.

Approach - Reduced DistilBert Transformer

To verify that the model was of the right complexity, we varied the number of attention layers and attention heads per layer. By default, DistilBert has 6 attention layers, each with 12 attention heads. We tried a variant of the DistilBert model with 2 attention layers and 6 attention heads each since we expected the model to overfit. We then trained this variant over 10 epochs instead of the 5 epochs used earlier. More epochs were required because changing the architecture would render the pretrained weights untuned.

7. Approach - POS Tags

We generate POS tags with nltk.pos_tag. The tags are then placed adjacent to the original token. A sentence like "Sun is bright" becomes "Sun ⟨NN⟩ is ⟨VB⟩ bright ⟨JJ⟩". We then feed these augmented sentences into the transformer. The embeddings are also expanded to include these tags as tokens. We expect the transformer to learn that these tags are associated with the word on its left. The model should then utilize them to better understand the structure of the sentence.

IV. Experiment and Result

Table I shows the experimental results of the 4 types of models and their variants with different pre-processing techniques applied.

For technique **balancing dataset**, the F1 scores shown in Table I are obtained by testing the models on the test dataset with the same internal ratio as the train dataset i.e. both the train dataset and the test dataset are balanced in terms of category. Table II shows the performances of logistic regression models and feed forword neural networks trained on balanced datasets of different sampling ratios when being tested using both balanced test dataset and test dataset following the original distribution i.e. the imbalanced test dataset.

V. Discussion

A. Analysis of the Performances of the General Pre-processing Techniques

1. Main Analysis Tool

In this project, we use Pointwise Mutual Information (PMI) and a metric derived from it called PMIScore to analyse how representative a word is to a category.

The calculation of PMI between a word and a category is as follows:

$$\mathrm{PMI}(word, category) = \frac{P(word, category)}{P(word) \times P(category)}$$

$$P(word, category) = \frac{df(word, category)}{\text{total number of headlines}}$$

$$P(word) = \frac{\sum_{c \text{ in categories}} df(word, c)}{\text{total number of headlines}}$$

$$P(category) = \frac{\text{number of headlines in } category}{\text{total number of headlines}}$$

where $df(x, c)$ is the document frequency of x in category c.

feature/model	Log Reg + TFIDF	Log Reg + Count Vect	Feed-forward NN	LSTM	LSTM + attention	DistilBert
tokenize (No stopwords)	0.7147	0.7412	0.773	0.7243	0.7498	0.7880
tokenize	0.7178	0.7412	0.779	0.7342	0.7524	0.7964
tokenize_bigram	0.7121	0.7484	0.759	0.7154	0.7400	0.7179
tokenize_synonym	0.6912	0.7040	0.674	0.5358	0.5558	0.7351
tokenzize_hypernym	0.5926	0.6749	0.552	0.2841	0.3262	0.6980
balancing dataset	-	0.790	0.782	0.7159	0.7441	0.8017
Synonym Augmentation	-	-	-	-	-	0.80803
Embeddings for Unknown Tokens	-	-	-	-	-	0.79463
Reduced DistilBert Transformer	-	-	-	-	-	0.7011
POS Tag	-	-	-	-	-	0.8275
Name Tagging	-	0.7012	-	-	-	-

Table I: F1 Scores (Macro) for the classifications using the models with different pre-processing techniques applied. As a comparison, the manual classification of 400 randomly sampled headlines has F1 Score (Weighted) of 0.848.

Count per Category in Train Dataset	Test Dataset	Log Reg + Count Vect	Feed-forward NN
2000	balanced	0.778	0.779
	imbalanced	0.736	0.740
4000	balanced	0.773	0.777
	imbalanced	0.739	0.753
10000	balanced	0.790	0.782
	imbalanced	0.740	0.757

Table II: F1 Scores (Macro) for the classifications using a few selected models trained on balanced datasets of different sampling ratios when being tested using both balanced and imbalanced test datasets.

The calculation of PMIScore of a word is as follows:

$$\text{PMIScore}(word) = \frac{\sum_{c \text{ in categories}} \text{PMI}(word, c)}{\sum_{c \text{ in categories}} 1 - I(word, c)} \times \left(1 + \sum_{c \text{ in categories}} I(word, c)\right) \quad (1)$$

$$I(word, category) = 1 \text{ if } \text{PMI}(word, category) = 0, \text{ otherwise } 0 \quad (2)$$

The intuition of PMIScore of a word is as follows: The higher PMI($word, category$) for a *word* with *category*, the more representative the *word* is to the *category*. If a *word* is perfectly representative of a *category*, then PMI(*word*, *category*) should be very high and 0 for all other categories. In the calculation of PMIScore, the mean of the PMI value of a *word* across the categories (i.e. the first component) would capture the PMI values, and the sum of the identity function I (i.e. the second component) would account for the presence of 0 PMI values. Hence, the higher the PMIScore of a *word*, the more useful it is in distinguishing the categories.

2. *Justification for the Performances*

From Figure 3 (left), we see that at least half of the stopwords have PMIScores above 0, and a number of them have very high PMIScores. Table III shows the PMI values and the PMIScore of the word with the lowest PMIScore among all stopwords. From the PMI values we can see that this word "show" is more tied to ENTERTAINMENT than any other category, and this property can make it useful in distinguishing the categories. Hence, keeping the stopwords in tokenization would let the models perform better than discarding the stopwords, a general trend shown in Table I.

Stopword	CRIME	RELIGION	TECH	MONEY	FOOD & DRINK
show	0.3477	0.7841	0.3062	0.2134	0.2194
SPORTS	TRAVEL	WOMEN	STYLE	ENTERTAINMENT	PMIScore
0.4662	0.3040	0.7046	1.172	2.280	0.680

Table III: PMI values between the word "show" to each category and its PMIScore

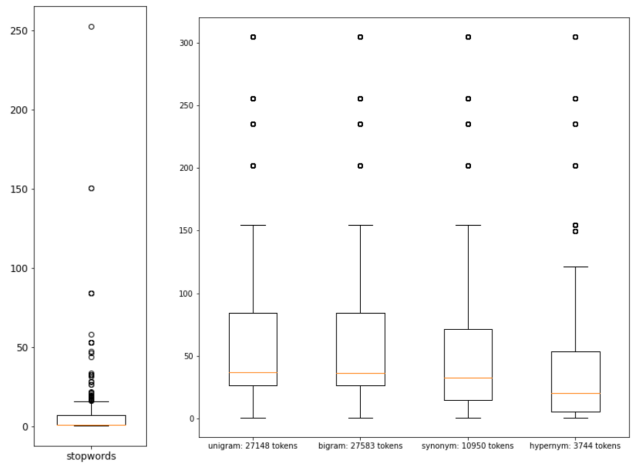

Figure 3: Box-and-whisker plots for PMIScores of Stopwords (left) and tokens generated using the 4 tokenization methods (right).

From Figure 3 (right), we can see that in general, the hypernym tokens and synonym tokens have the lowest and second lowest PMIScores. This lack of indicative power is also reflected in their worse performances as shown in Table I.

As shown in Table I, **balancing dataset** in general can boost the model performance. This boost of performance comes from upsamling the minority categories and keeping the majority categories intact. Otherwise, as shown in Table II, downsampling of the majority categories would lose the amount of information the models receive during training and be less effective in boosting the performance. We also need to take note that the categorization performances would only climax when the internal ratios of the train dataset and test dataset are similar, as reflected in Table II.

B. Logistic Regression

We are particularly interested in seeing if names play a significant role in the classification. This is because when we did classification by hand, most of us used names as

a gauge for certain categories with great accuracy.

From our findings in Table IV, we see a 1.5% decrease in f1 score when names are removed. However, there is little to no difference in f1 scores when location or organization tagged words are removed. From this, we can see that it is likely the names do play a role in the models ability to classify categories correctly.

Control	Names Removed	Organizations Removed	Location Removed
0.7412	0.7301 (-1.5%)	0.7392 (-0.2%)	0.7401 (-0.1%)

Table IV: F1 Scores when certain categories of words are removed

C. Studying the effects of Glove vectors

We had a hypothesis that removing word vectors similar to sentence vector decreases model performance. By comparing the cosine similarity of each word vector with the computed sentence vector, we attempted to only include the word vectors that were of a specified range. The results are given in Table V.

Cosine Similarity Range	0 - 1	0.1 - 1	0.2 - 1	0.3 - 1	0.4 - 1	0.5 - 1	0.6 - 1	0.7 - 1
F1 Score	0.777	0.777	0.777	0.775	0.763	0.726	0.596	0.344
Cosine Similarity Range	0 - 1	0 - 0.9	0 - 0.8	0 - 0.7	0 - 0.6	0 - 0.5	0 - 0.4	0 - 0.3
F1 Score	0.777	0.774	0.766	0.694	0.486	0.288	0.145	0.062

Table V: Cosine Similarity Range of Glove vectors included at test time vs F1 Score

From this, the data supports this hypothesis. Removing word vectors that are less representative of the sentence vector has a less drastic impact as compared to removing the word vectors that are more representative of the sentence vector. Keeping vectors of 0.3 - 1 yields a performance of 0.775 whereas keeping vectors of 0 - 0.7 yields a performance of 0.694.

disney	reveals	opening	seasons	for
0.581	0.429	0.542	0.573	0.496
star	wars	theme	park	lands
0.605	0.555	0.630	0.628	0.502

Table VI: Cosine similarity of the headline 'Disney reveals opening seasons for 'Star Wars' Theme Park Lands. ' against its sentence vector

From Table VI, we observe that when ranked by cosine similarity in descending order, the words appear in the following order: ["theme", "park", "star", "Disney", "seasons", "wars", "opening", "lands", "for", "reveals"]. For this example, the key words are closest to the sentence vector.

D. Synonyms for data augmentation

This approach was able to reduce overfitting as seen in Section VII.C. This augmentation is better able at retaining the semantics of the sentence whilst injecting randomness at train time. It was thus able to discourage overfitting whilst maintaining the F1-score as seen in Table I. It also performed better than the tokenize_synonym approach since the tokenize_synonym approach reduces variation of tokens across the categories leading to poorer classification results. The data augmentaiton approach however expands it by choosing synonyms randomly.

E. Embeddings for Unknown Tokens

This approach also led to no improvement. Further investigation found that the unknown tokens in the training set did not occur in the test set. Moreover, these unknown tokens also had a low frequency in the training set. The model was thus unable to learn anything meaningful about these new tokens.

F. Reduced DistilBert Transformer

From appendix, we observed a divergence between the training and validation loss as in Section VII.B, thus indicating overfitting. In addition, we also noticed a lower F1-score of 0.7011 on the test set as from Table I. As the simpler model was still overfitting and had a lower F1-score, we abandoned the approach.

G. Attention Mechanisms

We've noticed an improvement in the scores for models utilizing the attention mechanisms, as such we wanted to check the attention output to verify that the models are attending to the right tokens.

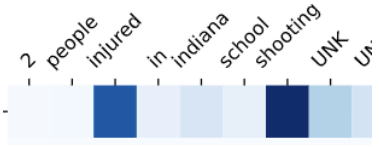

Figure 4: Attention Heatmap of LSTMAttention on CRIME headline.

From Figure 4, we have a heatmap of LSTMAttention on a CRIME headline "2 people injured in Indiana school shooting UNK". The attention layer here is attending to words relevant to the CRIME category. However, it seems to only be focusing on keywords. Such behaviour could make it predict wrong categories as it isn't fully understanding the sentence. We will observe this later.

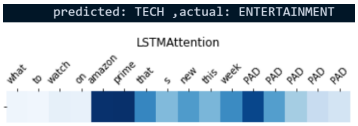

Figure 5: Attention Heatmap of LSTMAttention on headline

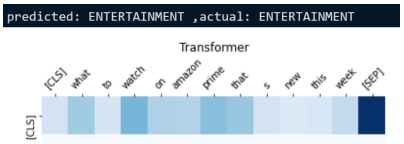

Figure 6: Attention Heatmap of DistilBert on headline

From Figure 5, we can see how LSTM Attention is only attending to keywords that are indicative of categories. Here the keywords, "amazon prime", which must have frequently occurred in the TECH category, were attended to wrongly. We can see that the LSTM's attention mechanism was not able to understand that the sentence's focus was on what content to watch on amazon prime as correctly identified by DistilBert in Figure 6.

H. DistilBert Error Analysis

To further improve our model, we evaluated all our test data and got inputs that produced the top K

losses. Upon manual inspection, we found that several of these inputs had either unknown tokens or word sense ambiguity, as such we decided to implement POS tagging as a feature.

I. POS Tags Analysis

1. *Word Sense Disambiguation*

Figure 7: Attention Heatmap of Transformer on headline

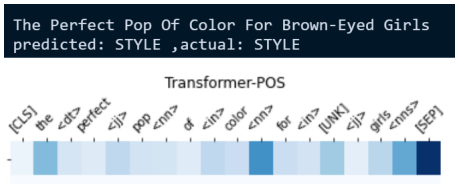

Figure 8: Attention Heatmap of TransformerPOS on headline

The above, Figure 7, does not use POS tags while the below, Figure 8, does. Here we note that the transformer without POS tags makes the wrong prediction. From the heatmap Figure 7, we see that it attends to the object "pop" instead of "color" even though "pop" is the indirect object and "color" is the direct object. With POS tagging, the model is better able to disambiguate the meaning of "pop" and "color" when provided with the right POS tags. With the POS tags, the model has learnt to attend to "color" more than "pop" as seen in Figure 8 and thus determine correctly that this is a STYLE headline and not a WOMEN one.

2. *Unknown token clarification*

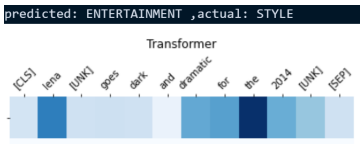

Figure 9: Attention Heatmap of Transformer on headline

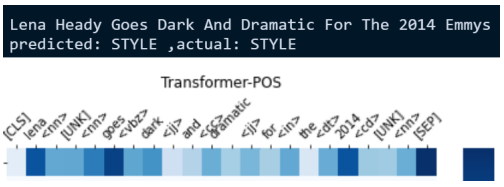

Figure 10: Attention Heatmap of TransformerPOS on headline

The above, Figure 9, does not use POS tags while the below, Figure 10, does. Here we note that the transformer with POS outperforms. For this example, we notice 2 UNK tokens. With POS tagging, the model is able to gather that the first UNK token, UNK_1, is a noun. With this information, it is able to determine that the phrase, "lena UNK goes dark and dramatic", is the theme of the sentence. The POS tag could have helped the model learn that "lena UNK" is a name with the help of the noun tag for UNK_1. Without the POS tags, the model is unable to determine that "lena UNK goes dark and dramatic" is a coherent phase and thus wrongly attends to the later part of the sentence. As such, it wrongly predicts it as an ENTERTAINMENT headline.

WordCloud for CRIME category

Apart from individual examples, we've also noticed that in general, the tokens attended to were more relevant to the category. Here is just one example of the tokens attended to from the CRIME category with POS tagging.

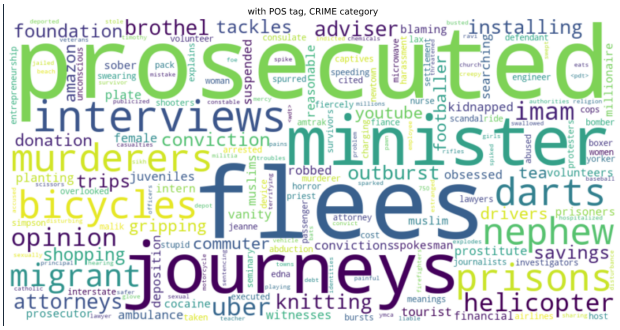

Figure 11: WordCloud of attention from TransformerPOS on CRIME headlines

Figure 11 shows a WordCloud generated from the average attention a token receives across the dataset. We can see that the transformer with POS tagging has attended to relevant tokens. For the WordCloud of the transformer with no POS tagging, please see Section VII.D.

The fact that tags were not heavily attended to is actually a relief. This suggests that the model was using them as context clues and not blindly memorizing tags as keywords indicative of a particular category.

Shortfall of POS tagging

The tags were beneficial when they tagged data accurately, there were unfortunately sentences that were not properly tagged. This is a shortfall that we could address as an extension of the project.

VI. Conclusion

To conclude, this paper explored the use of different models in the problem of news headline classification. We explored the most basic model of logistic regression and moved to more complex models like Feed-forward Neural Networks, RNNs and Transformers. Of these models explored, DistilBert gave the best f1-score (macro) of 0.8275.

Moving forward, more categories can be added. Further training and research has to be done to better improve the classification. More computing power is required for training with bigger data sets as well.

References

[1] Jason Brownlee. *Parametric and Nonparametric Machine Learning Algorithms*. Aug. 2020. URL: https://machinelearningmastery.com/parametric-and-nonparametric-machine-learning-algorithms/.

[2] Ian T. Jolliffe and Jorge Cadima. "Principal component analysis: a review and recent developments". In: (2016).

[3] Rishabh Misra. *News Category Dataset*. Dec. 2018. URL: https://www.kaggle.com/rmisra/news-category-dataset.

[4] M. Thangaraj and M Sivakami. "Text Classification Techniques: A Literature Review". In: *Interdisciplinary Journal of Information, Knowledge, and Management* 13 (2018), pp. 117–135. DOI: 10.28945/4066.

VII. Appendix

A. DistilBert Transformer Overfitting

From Figure 12 and 13, we can see that the training loss nears 0 but validation loss is stuck at 0.5. This shows clear evidence of overfitting to the training data.

B. Small DistilBert Transformer Overfitting

From Figure 14 and 15, we can see that the training loss is much lower than the validation loss. This shows clear evidence of overfitting to the training data.

C. Data Augmentation

From Figure 16 and 17, we can see that the data augmentation reduced overfitting. Specifically, validation loss has decreased and training loss has increased. The divergence between training loss and validation loss has thus been reduced with this technique.

D. Transformer vs TransformerPOS WordCloud on CRIME

From Figure 18 and 19, it is clear that POS tagging has helped the Transformer attend to more relevant tokens like prosecuted as opposed to internet.

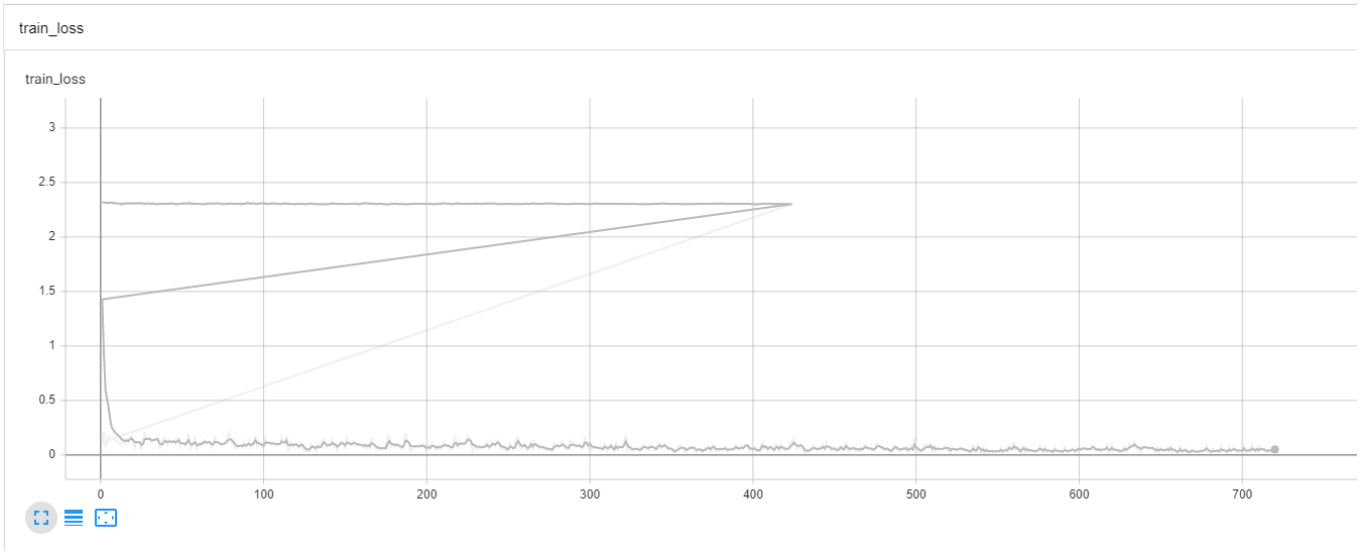

Figure 12: Plot of training loss of DistilBert

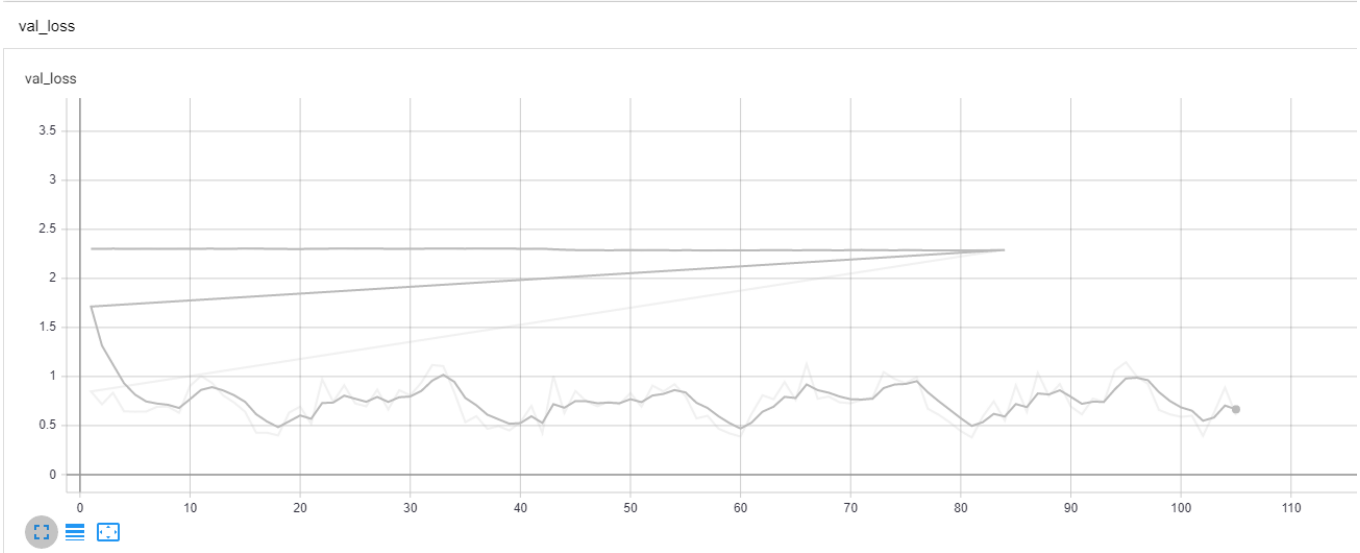

Figure 13: Plot of validation loss of DistilBert

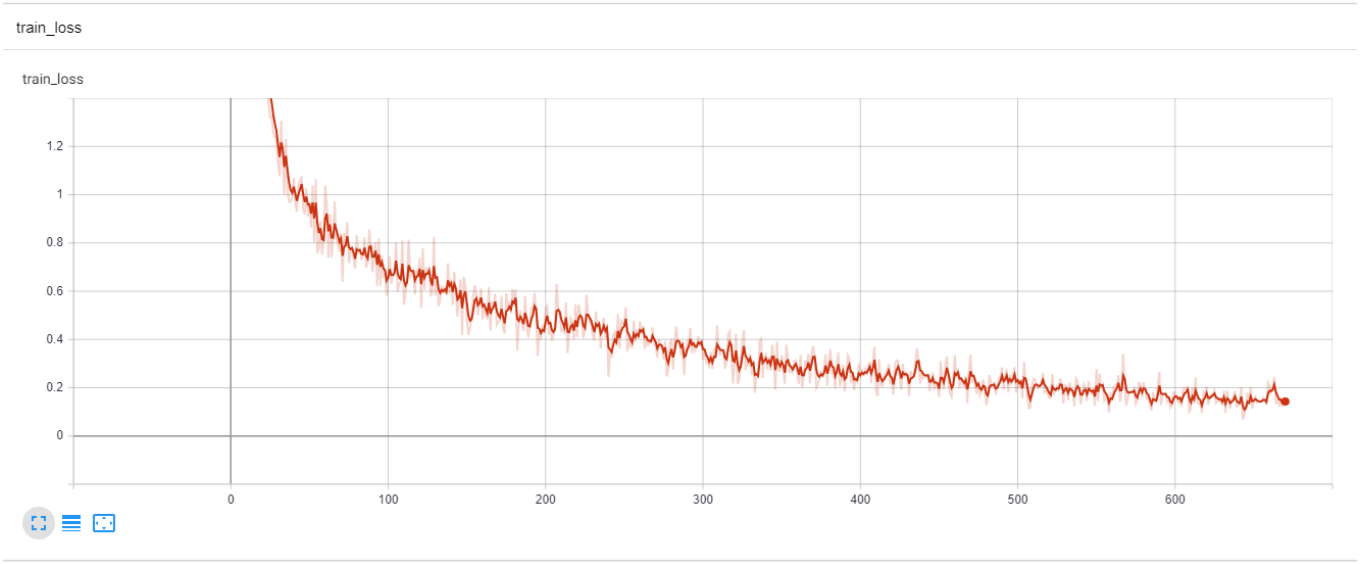

Figure 14: Plot of training loss of Small DistilBert

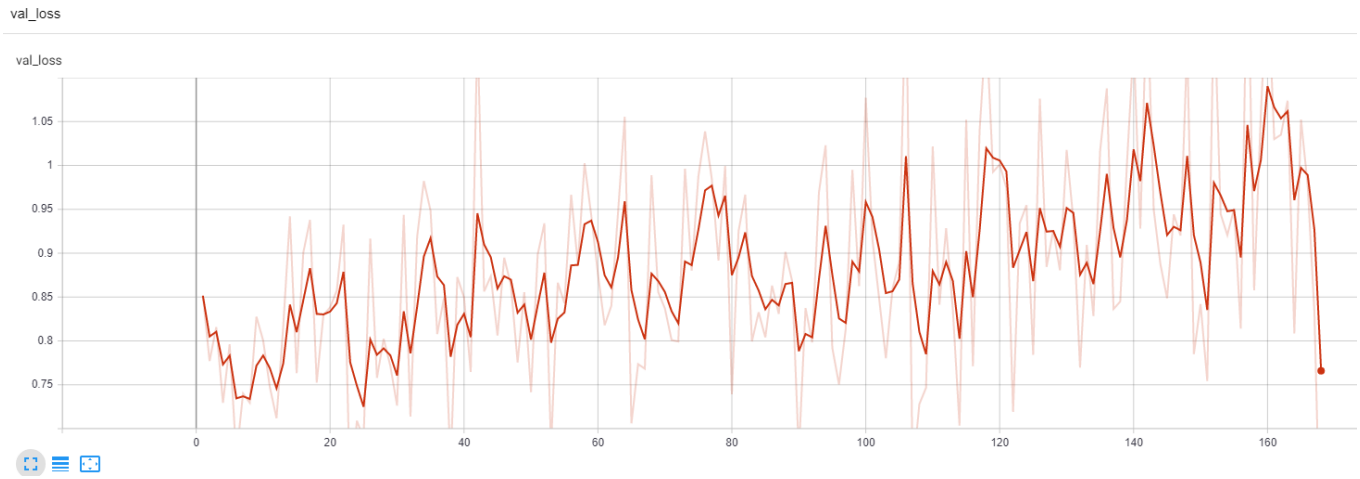

Figure 15: Plot of validation loss of Small DistilBert

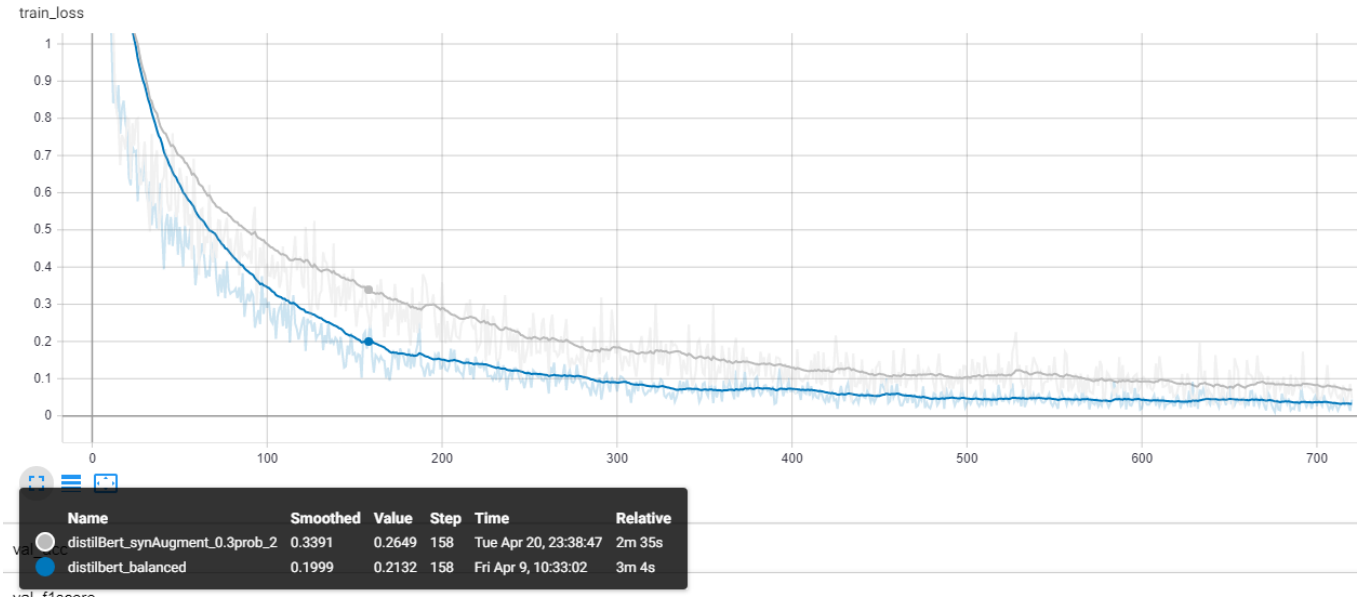

Figure 16: Plot of training loss of DistilBert + synonym augmentation (Grey is with augmentation, blue is without)

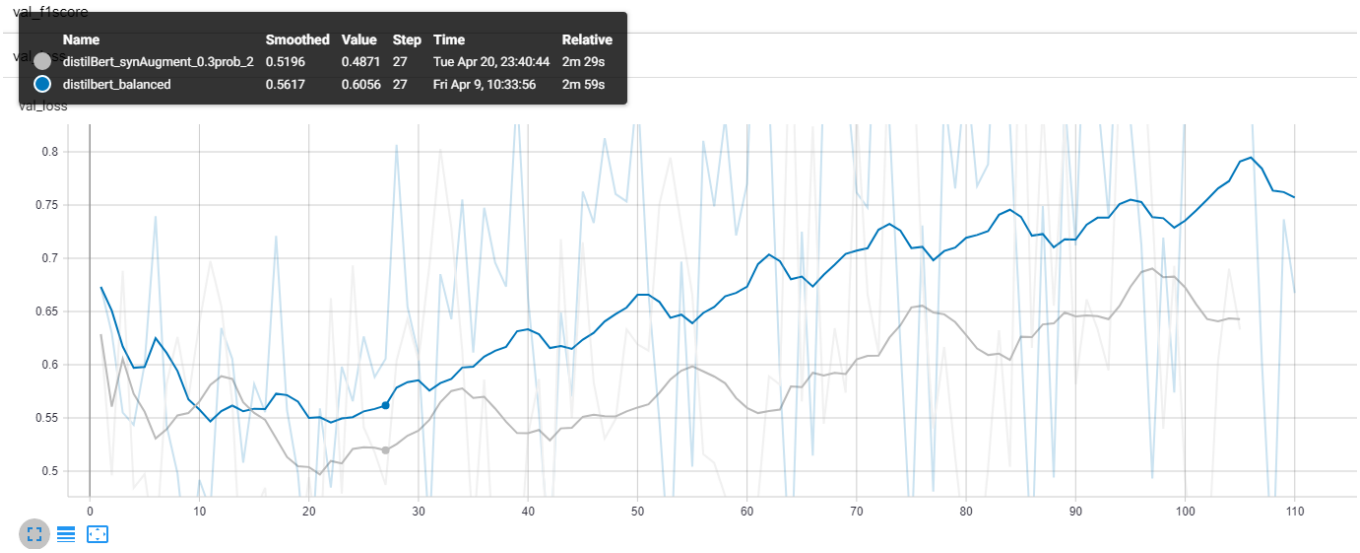

Figure 17: Plot of validation loss of DistilBert + synonym augmentation (Grey is with augmentation, blue is without)

Figure 18: WordCloud of attention by transformer on CRIME headlines

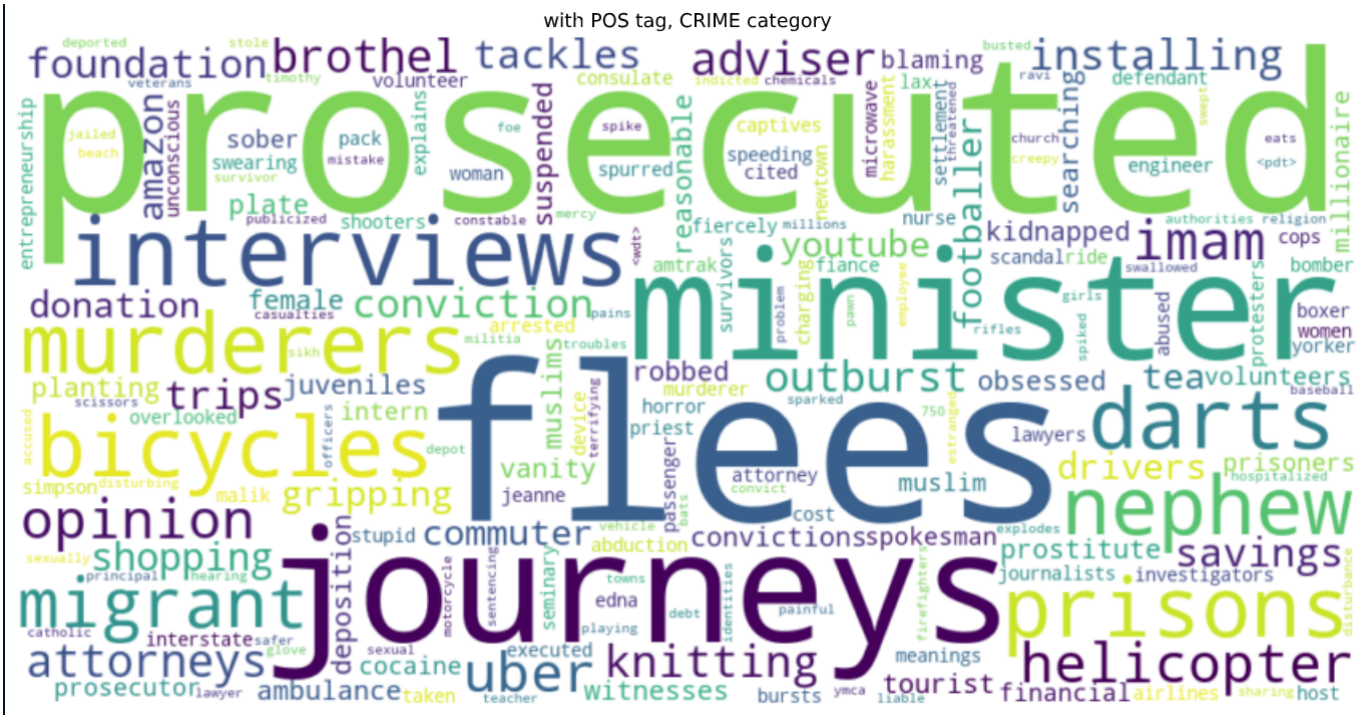

Figure 19: WordCloud of attention by transformer + POS on CRIME headlines