Natural Language Processing Assignment

Michael Simpson

EEECS
Queen's University Belfast
Northern Ireland
2nd March 2025
Word Count: 2885

Introduction

The ability for humans to understand textual information comes almost as naturally as speaking, but few are probably aware of how complex and deep this understanding goes. Humans are capable of reading paragraphs of texts, all while understanding the intricate grammer within sentences and the meaning of many different words, as well as create links between different sentences and paragraphs to derive contextual meaning within a large body of text. Hence, one can see why it has been a long and arduous process to "teach" machines how to fully understand pieces of text as well, beginning back in the 1940s.[1] In the current age, one can easily train many different types complex models on pieces of text using common programming languages, to extract lots of useful information, depending on the users preference. This was essentially the goal of the following project.

I was provided with 600 short stories that were generated using OpenAI's GPT-4 model. Each story was generated from metadata such as the story theme and setting, as well as the character involved in the story. An example of one of the stories is given below, along with all the metadata. The first part of the project looked at the length of dependencies within the stories to investigate the change between stories of different styles. The frequencies of POS tags were also found for each story and whether they change for different kinds of stories. The second part involved the classification of story text by story 'setting' using 2 different models: Word2Vec and BERT. The third and final part was analysis of sentiment using BERT i.e. classifying the stories by outcome (victory or defeat), and also an investigation into the emotional content of a story using a pre-trained emotion detection model.

Section 1: Text processing and linguistic insights

Writing style and dependency parsing

To generate each story, the prompt included 1 of 5 writing styles, which are shown in Figure 2 alongside a description of the style. The difference in linguistic strucure of these styles were investigated by measuring the dependency lengths for each story. A dependency within a sentence occurs whenever a word is related to another word in the sentence by grammatical context. An example might be "The cat eats fish", where the word "cat" is related to the word "fish" through the verb "eats", which is the root. In this example, the length of the dependency is 2, as 2 steps are required to get between the words. In general, the dependency length can be used as a measure of the syntactic complexity of a sentence. Therefore, if the dependency lengths were found all the stories, then one can investigate whether they change significantly between stories of different styles.

To obtain the dependency lengths from a piece of text, a function was created that firstly processed the full story text using *spacy*, which automatically

```
{'index': 256,
 'theme': 'redemption',
 'setting': 'outer space',
 'person': 'spy',
 'style': 'descriptive',
 'outcome': 'defeat',
 'object_concept': 'cabbage',
 'story': "Among the cold, shimmering cosmos, as distant and indifferent as the
ancient gods, there existed a woman of perplexing skill and obscurity.
Enrobed in the cloak of constellations, she made her living on the precipice of
ambiguity and secrecy, artfully navigating the labyrinthine corridors of subterfuge
and manipulation. She was a whisper upon the lips of the unknown, a spirit who
danced blithely among the shadows of the endless cosmic expanse.\\n\\nA task of paramount
importance lay before her, a mission that carried the weight of countless lives.......",
 'question1': 'What item does the protagonist carry with her and what is its purpose?',
 'answer1': '"She alighted on desolate asteroids, wistful fragments of forgotten worlds,
carrying with her a singular, peculiar possession: an unassuming cabbage,
its verdant leaves concealing an arcane technology capable of ensnaring the astral entity."',
 'question2': "What is the outcome of the protagonist's mission?",
 'answer2': '"The entity, in all its incandescent chaos, could not be tamed or controlled,
could not be chained by the clandestine desires of a woman who walked the
path of shadow and subterfuge. Her gambit failed, the cabbage\'s hidden magic overwhelmed
and short-circuited, leaving her floating aimlessly amid the endless tapestry
of the cosmic theater, a single horn sang her swan song, echoing a haunting ballad of
failure and lost intentions."'}
```


Figure 1: An example of one of the stories generated by OpenAI's GPT-4 along with the metadata involved to create it. The story is an extract as the full text is too long to show

"descriptive": Long sentences, rich modifiers.
"concise": Shorter sentences, fewer modifiers.
"poetic": Complex sentence structures, metaphorical language.
"journalistic": Neutral, declarative sentences with medium complexity.
"for children": Simple sentence structures, simple vocabulary.

Figure 2: The five writing styles used as part of the prompt for the language model to generate the stories.

retrieves NLP information about the text. Each sentence was then iterated through and also tokenised which allowed for the head of each token to be found, where the head is the word that the token is related to. Recording the index of the token and it's head meant the dependency length of each token could be found, and as a result the largest length for that particular sentence. The average of these maximum dependency lengths were then returned for each story from the function. The JSON file containing the stories was iterated through to send each story text to the function.

```
#Load the english model
nlp = spacy.load("en_core_web_sm")
#Process the text
doc = nlp(text)
```

Figure 3: This code was used to process each story text as it passed through the function

A Pandas DataFrame was then created which contained the index of the story, the story style and the Average Maximum Dependency length. The stories were grouped by style and the average of the dependency lengths were calculated, which can be found in the table below. Because an average calculation might not represent the data very well, a raincloud plot was also made.

Story Style	Avg. Max. Dependency Length
Descriptive	28.0
Poetic	23.5
Journalistic	18.7
For Children	13.5
Concise	6.76

Table 1: The Average Maximum Dependency Lengths for each story style

It is clear to see from the table and the raincloud plot that the "descriptive" writing style produced stories with the largest dependencies, and are thus syntactically complex, whereas the "concise" style had very short dependencies. In fact, the raincloud plot shows that there was no overlap between the dependency lengths between the two styles, indicating stark contrast in complexity. This was expected due to the description of each style being in contrast to each other, as shown in Figure 2. The contrast is less clear, however, for the other styles, especially between "journalistic", "poetic" and "descriptive", as there is alot of overlap in dependency lengths. The average for "descriptive" also appears to be larger due to a few stories with exceptionally long dependencies. Therefore, the difference in complexity of the linguistic structure for these story styles can only minimally be ascribed to the difference in dependency lengths, and so a different metric would have to be used. In other words, "descriptive" and "poetic" are clearly very different styles, but the dependency length appears to indicate that they are similar.

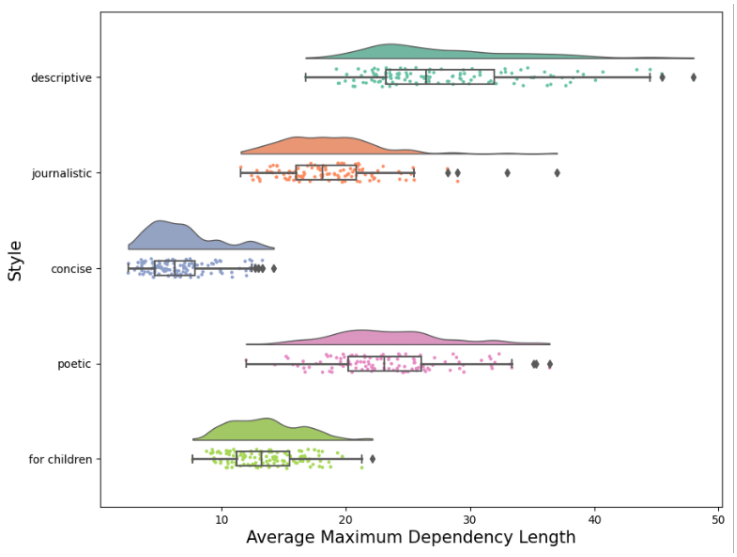

Figure 4: A Raincloud plot showing how dependency length differs between story writing styles.

POS tagging

A POS (Parts of speech) tag is label assigned to words to indicate it's grammatical role within the text. Examples of POS tags include Nouns, Verbs and Adjectives, as well as variations of these such as Nouns Plural. Counting the frequencies of these for a piece of text can help with identifying key linguistic differences between other pieces of text, and in our case, between story styles.

To count the frequency of the POS tags, a very similar function was used to the one in the previous section, but the *_pos* was used to obtain the tag for each token. The count of each tag was then returned from the function. This was done for each story by iterating through them. Similar to the previous section, the count of tags for each story was grouped by style, then the mean count of each tag was calculated. The bar chart below shows 5 POS tags and how their frequency differs between story styles.

Across all 5 POS tags shown, it is clear that the "descriptive" stories contain the highest frequency of them all, except for PRON. This makes sense as those stories are the longest, and thus contain the most number of words. The "for children" style contains the most PRON tags which could be explained by that the fact that children's stories tend to involve a character with continous reference to this character. It is interesting to note that for all the tags except PRON, the distribution is roughly the same which indicate that these tags do not differ between types of stories.

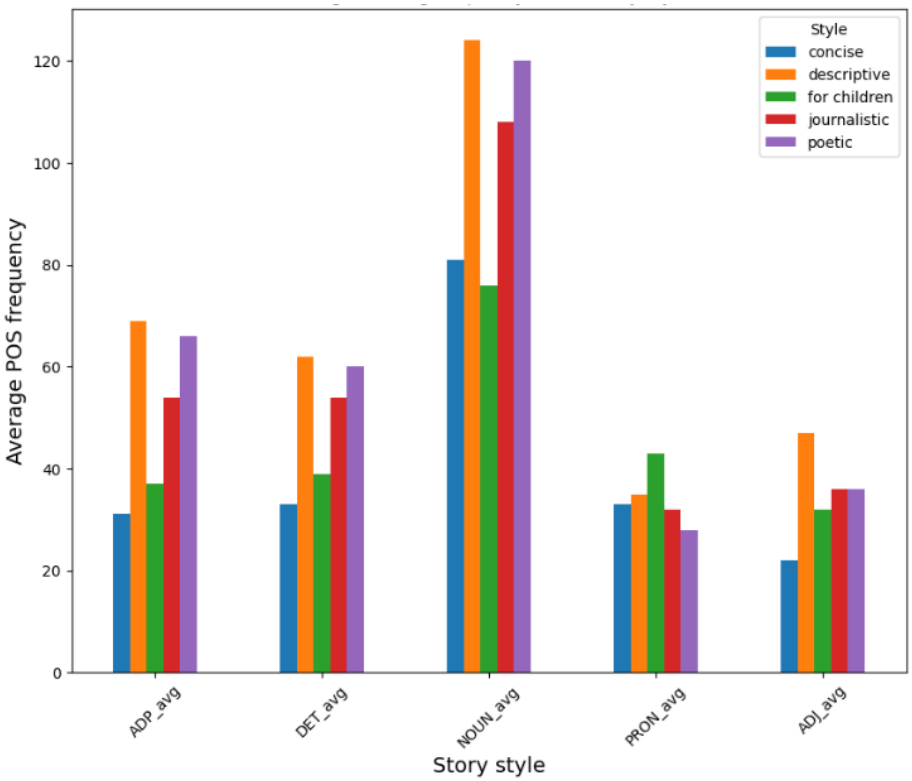

Figure 5: The frequency of 5 POS tags between the story styles. ADP = Adpostion, DET = Determiner, NOUN = Noun, PRON = Pronoun, ADJ = Adjective

Each story was also generated with the outcome being "victory" or "defeat", so it was suspected that certain POS tags would appear more in one than the other. The suspected tags included verbs, adjectives, adverbs, nouns and determiners. As an example, verbs were suspected to appear in victory stories due to the appearence of words such as won, defeated or achieved. The dataframe obtained before was split in two: one for the victory stories, and the other for the defeat stories. An unpaired t-test was then performed, setting the p-value threshold at 0.05, between the two dataframes for each suspected POS tags. Below are the results of this.

Interestingly, the p-values calculated for each of the 5 POS tags were much

	t-statistic	p-value
VERB	0.812	0.417
ADJ	-0.157	0.876
ADV	1.36	0.174
NOUN	0.623	0.533
DET	1.10	0.272

Table 2: The t-statistic and p-values calculated for each POS tag between stories with outcome victory and defeat

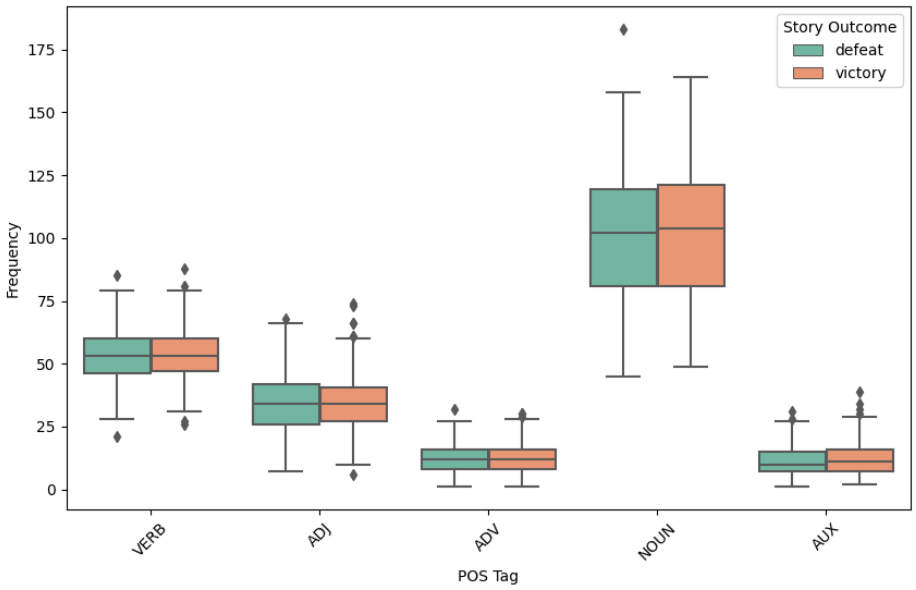

Figure 6: The frequency of the 5 suspect POS tags for both stories of outcome victory and defeat.

larger than the set threshold of 0.05. This indicates that any significant different in the frequencies of the tags is simply due to random statistics, rather than any meaningful difference. Figure 6 also clearly explains the large p-values, since all the box plots for each POS tag are of similar shape and position, showing that there is an approximately equal number of each POS tag between the types of stories.

Section 2: Text Classification

Word2Vec

Another piece of of metadata provided to the model for generating the stories was setting, with the example in Figure 1 being "Outer Space". The other possible settings for each story were "Post-apocalyptic world", "Small town", "Ancient Civilisation" and "Modern Metropolis". This section then dealt with the training of a model such that it could predict the setting given the story text.

The first model used was Word2Vec, which is capable of converting words into numerical vectors that capture their meaning based on context. The model was pre-trained using both a Bag-of-Words technique, where it predicts a word from it's surrounding words, and Skip-Gram, which predicts the surrounding words from a target word. A word is therefore "close" in meaning to another word if both their vectors are near each other in this high-dimensional space.

The model was downloaded and a function was created that tokenised the text for each story, then retrieved the vector for each token and added it to a list. The feature vector for each story was then obtained by averaging across all of the word vectors for a particular story.

The feature vectors and a list of the settings were split into training and test sets (80% training and 20% test). A Linear SVC model was then used by firstly training it then making predictions for the settings. The accuracy for this came out to be 84.3%. The classification report for this can be found below as well.

```
vector_list.append(model.get_vector(token))
```

Figure 7: The "get_vector" method is used as part of the Word2Vec model to obtain vectors for words.

Setting	Precision	Recall	F1-Score	Support
A Modern Metropolis	0.96	0.83	0.89	29
Ancient Civilisation	0.84	0.76	0.80	21
Outer Space	0.89	0.94	0.91	33
Post-Apocalyptic World	0.67	0.78	0.72	18
Small Town	0.81	0.85	0.83	20

Table 3: Classification report for the Linear SVC model predicting story setting from the test set.

Clearly, using Word2Vec along with Linear SVC is an effective way of predicting the setting of these stories. However, this was to be expecting as the settings are very distinct and thus would contain very different word vectors, allowing the model to generalise easily. It would be interesting to investigate how

the same model pipeline would perform if the choice of settings were different and less distinct.

BERT

BERT is a deep-learning model that is based off of the transformer architure and stands for Bidirectional Encoder Representations from Transformers. The model considers full sentence context by processing the text from left to right and right to left i.e. bidirectionally. This is unlike Word2Vec which looks at the surrounding words one-way. In training, the model also masks some of the words in a piece of text then predicts what the word is. It is very powerful in question answering and sentiment analysis, but it was used for the purpose of setting classification in this section.

Some data pre-processing had to be done before applying the BERT model. Firstly, the story with the largest number of words was found which came out to be 869. BERT can input text with a maximum of 512 so the maximum length was set to this number. This meant that all stories with total word count being larger or smaller than this were then truncated or padded, respectively. The stories were split into training, validation and a test set before being passed through a class which dealt with encoding the story text, returning the attention masks and also transforming the target labels (settings) into tensor format. The datasets created from this class were then made into dataloaders which allowed for the data to be split into batches for training. Finally, the BERT model was loaded in from the HuggingFace library and turned in a classifier by adding a linear layer and by specifying the number of classes being five.

To achieve the desired results the AdamW optimiser was used which is better for generalisation by reducing overfitting. The learning rate could also be changed within this optimiser and was chosen to be 2.5e-5. It was changed a few times, ranging from 1e-5 to 4e-5 as the outcome of BERT seemed to be quite sensitive to this. A scheduler was also created that meant the learning rate could change during training every certain number of steps. The training data was then passed through the model to train while also passing through the validation data during each epoch. The results of this process can be seen below.

Epoch	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
1	1.69	0.185	1.62	0.25
2	1.48	0.308	1.15	0.4
3	0.716	0.807	0.326	0.95
4	0.184	0.981	0.150	0.95

Table 4: The training and validation losses and accuracies for the setting classification using BERT over 4 epochs.

The best BERT model was then saved and used to classify the test data

with a test loss of 0.198 and accuracy of 95.1%. Both the classification report and confusion matrix can be found below as well.

Setting	Precision	Recall	F1-Score	Support
A Modern Metropolis	1.00	1.00	1.00	16
Ancient Civilisation	0.85	1.00	0.92	11
Outer Space	1.00	1.00	0.83	18
Post-apocalyptic World	0.83	0.83	0.83	6
Small Town	1.00	0.80	0.89	10

Table 5: Classification report for the BERT model classifying story setting

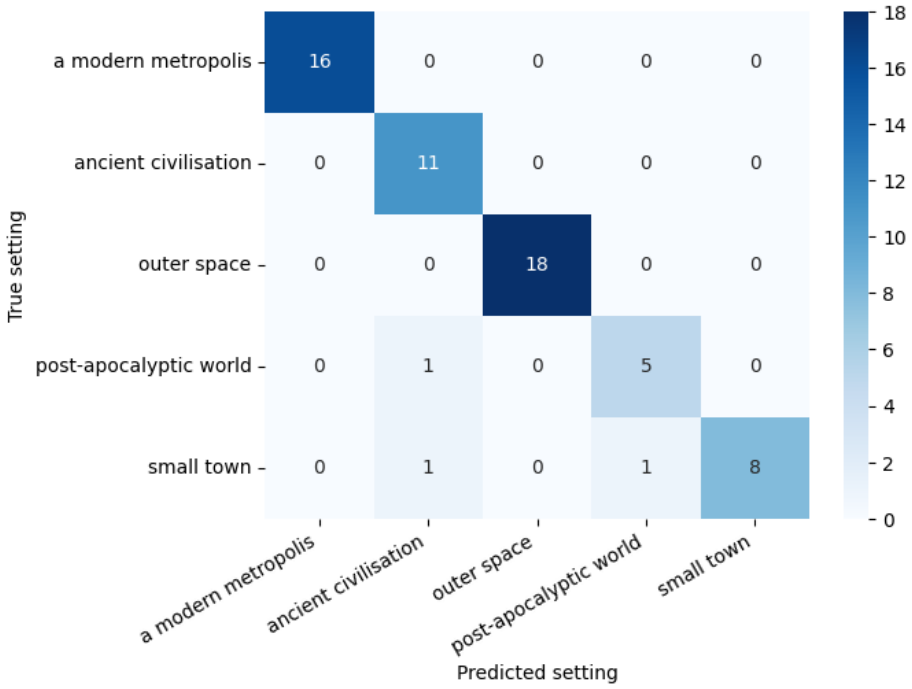

Figure 8: Confusion matrix showing the performance of the BERT model when classifying story settings from the test dataset

This was an expected result from BERT since the transformer architecture is currently state of the art, being used as part of Open AI's GPT models. Again, this model would be interesting to test on stories with settings that are not so distinct, and then comparing the results to Word2Vec.

Section 3: Sentiment and Emotion

Sentiment analysis with BERT

BERT is not only great for textual classification, but also for detection of sentiment within a piece of text, such as the general mood (happy, neutral or sad). In the case of the stories, each one either ends in victory or defeat, which clearly invoke two seperate kinds of moods if a human were to read them. In this section, BERT was trained to detect this sentiment from only the last sentence of each story.

```
Victory story:
The skilled creator, once unacknowledged, stood vindicated in the heart of the city,
his honour restored, his brilliance undeniably resplendent amidst the mocking echoes
of the concrete jungle.

Defeat story:
He was defeated, but the echo of his trombone continues to resonate in the silent nights,
reminding the community of the dreams that failed to see the dawn.
```

Figure 9: The last sentence of two stories, one for victory and one for defeat. It is easy to detect the sentiment of the stories just from their last sentences.

To obtain the last sentence of each story, a sentence tokeniser was used and the last sentence was simply selected from the list and and added to its own seperate list. This list was then used to create a new column in the stories dataframe containing the last sentence of each story. This meant that the exact same method used in 2.2 could be applied here where the "last sentence" column comprised the input data, and the "outcome" column was the target labels. The maximum length was set 116, as this was the number of words in the last sentence found to be the longest. The BERT classifier was also slightly modified to have 2 classes, for victory and defeat. Training the model then applying it to the test set produced the results shown below, and with a test loss of 1.04 and accuracy of 77%.

Epoch	Training Loss	Training accuracy	Validation Loss	Validation Accuracy
1	0.68	0.547	0.653	0.633
2	0.308	0.863	0.562	0.783
3	0.0967	0.965	0.567	0.85
4	0.00854	0.998	0.694	0.85

Table 6: The training and validation losses and accuracies for sentiment analysis using BERT.

With a 77% accuracy, the BERT model performed fairly well at detecting sentiment, given that only the last sentence of each story was provided. Due to the presence of False Negatives and Positives, however, the model is clearly

Outcome	Precision	Recall	F1-Score	Support
Defeat	0.83	0.74	0.78	34
Victory	0.71	0.81	0.76	27

Table 7: Classification report for the BERT model classifying story outcome

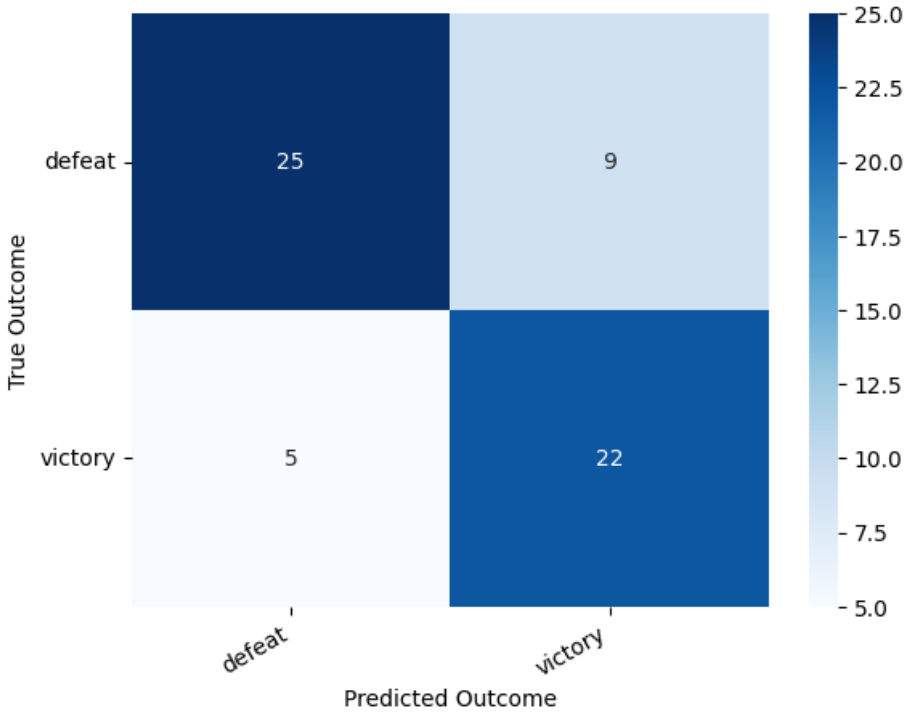

Figure 10: Confusion matrix showing how BERT performed when detecting story sentiment i.e. outcome classification

missing some important context that would be given from the rest of each story. It can be reasonably hypothesised that if the full story texts were given, the accuracy would be much higher.

Emotional content of stories

This last section dealt with the detection of emotions in the stories using an emotion detection model from the HuggingFace library. The model used was Roberta for multi-label classification.[2] The model works by processing a sentence then outputting a list of emotions and their scores, where higher scores correspond to the emotions most likely associated with that sentence. There-

fore, to obtain the emotion for each story, a sentence tokeniser was used and the emotions and their scores were found using the model. A function was then created that took all the scores for each sentence and averaged them to obtain the emotion scores for each story. The emotion with the highest score for each story was added to a list to create a new column in the story dataframe with these emotions. To test the effectiveness of the model, the stories were grouped by theme ("Betrayal", "Discovery", "Love", "Rebellion" and "Redemption"), then a bar chart was created showing the distribution of emotion within each theme.

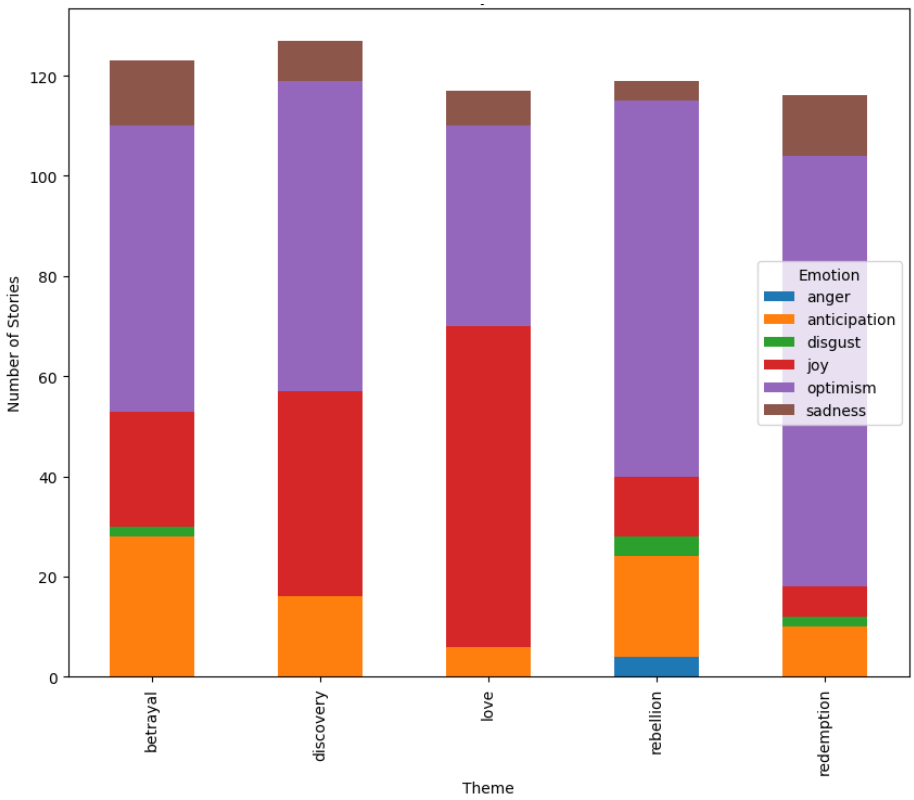

Figure 11: A bar chart showing the distribution of emotion detected by Roberta for each story theme.

The dominating emotion detected by the model is clearly optimism for most themes except for "love". If one thinks about stories generated using these themes, it is easy to imagine elements of each story containing optimistic words and phrases, especially for the theme of "redemption", which will often be about characters starting at a low point then progressing towards being a better version of themselves. While the theme of "love" also had elements of optimism

detected, joy is dominating and it isn't difficult to imagine why that is the case. Therefore, it can be concluded that the model has a strong ability to detect emotion, but it would be interesting to test on stories with different themes that may lean towards more negative emotions such as sadness and disgust.

Conclusion

This project has been an investigation into various NLP models and how they can be applied to text, specifically, stories generated by Open AI's GPT-4 model. The stories generated had a wide range in style, theme, outcome and setting, providing the models with many elements they could use to generalise when making predictions. The first section found that "descriptive" stories tend to have longer dependencies with the opposite being true for "concise" stories, but this line became more blurred for the other writing styles. It was also found that the distribution of the frequency of POS tags did not change drastically between stories of different writing styles, and also of different outcomes, as evidenced by the p-values calculated. Section 2 dealt with the application of 2 models; Word2Vec and BERT to text classification, specifically, the classification of stories by their setting. BERT performed better than Word2Vec with an accuracy of 95.1%, which was expected due to the nature of transformers. The final section showed the application of a similar BERT model to sentiment analysis i.e. classifying a story by its outcome, with a final accuracy 77%. This accuracy was not too surprising since BERT had only the last sentence of each story to train on. Finally, a Roberta model was used to detect the emotional content of the stories, finding that "optimism" was dominant across all themes except for love.

This investigation clearly showed that these models, especially BERT, are extremely effective for performing a variety of different tasks. With more data to train on and fine-tuning, it is easy to see how the models could even surpass humans when it comes to understanding textual information.

References

[1] Jones, K.S. (1994) *Natural Language Processing: A Historical Review. In: Zampolli, A., Calzolari, N., Palmer, M. (eds) Current Issues in Computational Linguistics: In Honour of Don Walker. Linguistica Computazionale*, vol 9. Springer, Dordrecht.

[2] @inproceedingscamacho-collados-etal-2022-tweetnlp, title=TweetNLP: Cutting-Edge Natural Language Processing for Social Media, author=Camacho-Collados, Jose and Rezaee, Kiamehr and Riahi, Talayeh and Ushio, Asahi and Loureiro, Daniel and Antypas, Dimosthenis and Boisson, Joanne and Espinosa-Anke, Luis and Liu, Fangyu and Martínez-Cámara, Eugenio and others, booktitle = "Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing:

System Demonstrations", month = nov, year = "2022", address = "Abu Dhabi, U.A.E.", publisher = "Association for Computational Linguistics",