Final Report

Preface:

Our goal is to state the relevance of a search phrase to a corresponding item, this can help to rank the results of a search engine for Home-Depot's items.

The task at hand involves deriving a relevance score between a given search term and a corresponding product title for the product yielded by the search.

The relevance score labels provided take into consideration the semantic relation between both inputs and aren't just based on word similarity.

We tackle this task using 2 different approaches: character level processing and word level processing.

In each approach we will embed the text input and use the embedded representation in the construction of a naïve basic benchmark for the character level, and while constructing a Siamese Network for both approaches. We will use the output of said Siamese Networks for feature extraction purposes to be used on 2 machine learning models.

In the naïve benchmark we attempted to use a count vectorizer to create a matrix of all the character appearances in each sentence and calculate their similarity using the 'cosine similarity' algorithm. This proved a relatively poor benchmark seeing as there was no way to enforce semantic understanding of the given text, but rather only character similarities.

The Siamese Network propagated both embedded representations through the same LSTM layer which produced an output for each of the two inputs. Afterwards, we computed the difference between both outputs represented by vector arrays using cosine similarity. We also tried using a Euclidean distance measurement which we found to be lacking. We compared the result of the cosine similarity to a normalized relevance score (range 0-1) where 0 is the best relevance score and 1 is the worst.

We decided to utilize the XGBoost and LightGBM machine learning models, as these models are both gradient boosting frameworks that are well known for their performance as similarity rankers.

The results we acquired were better than the results we had for our naïve benchmark, but not as good as the network we constructed. We attribute this to the fact that the LSTM already returned a good representation vector for a certain window, therefore, the machine learning algorithms were overprocessing data that already provided a good result for our task.

Tokenization examples:

For tokenization, in both word-level and character-level we removed all conjunction words from the product title and the search term.

The conjunction words we removed were:

‘is’, ‘in’ ,’and’, ‘for’, 'or', 'yet', 'so', 'nor', 'both', 'whether', 'but', 'also', 'either', 'neither', 'just', 'the', 'as', 'then', 'than', 'such', 'that', 'after', 'although', 'much', 'by', 'inasmuch', 'even', 'when', 'where', 'since', 'because', 'whose', 'before', 'until', 'how', 'like', 'i', ’want’, 'a', 'an'.

We opted out of removing the words ‘no’ and ‘not’ because we found that it could assist the model rather than hinder it.

We decided to remove those words because we understood that they serve no purpose in the product search other than for easier and streamlined human interaction.

In addition, we split the sentences into words with the following splitting terms: ‘/n’, ’ ‘, ‘;’, ’_’, ’+’, ’-‘, ’,’.

We did so to split sequences that describe more than a single word such as “Gray-DISCONTINUED” into 2 words.

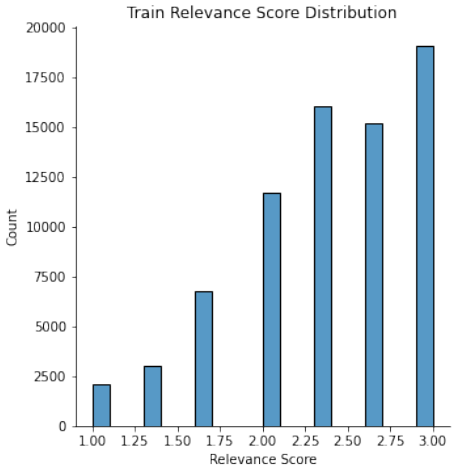
The reason being because we noticed the amount of words in each search term is small and lacking details. It is more probable to find the word “Gray“ and not “Gray-DISCONTINUED”.

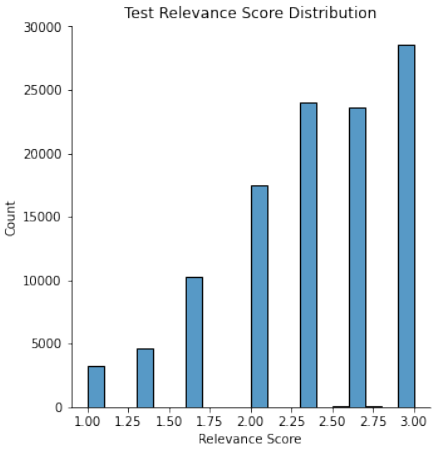
Exploratory Data Analysis Plots:

Display the relevance score distribution:

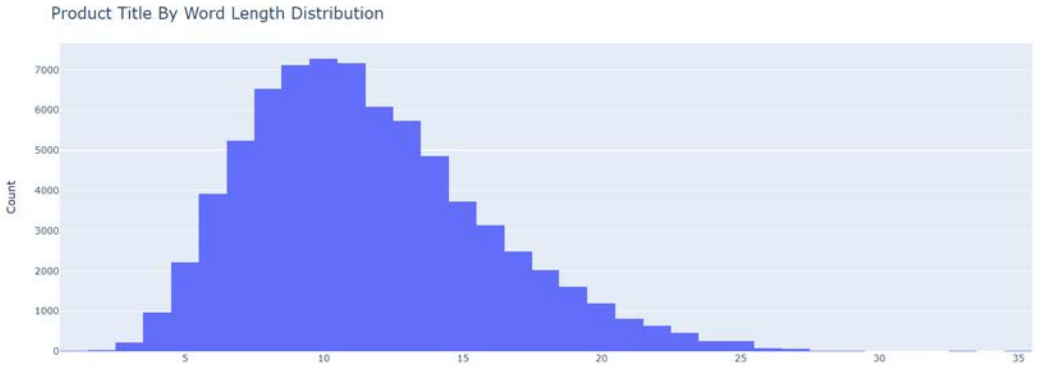
We can see that our data contains many examples that fits the search terms provided, therefore, the engine that receives the search terms will most likely succeed in its search.

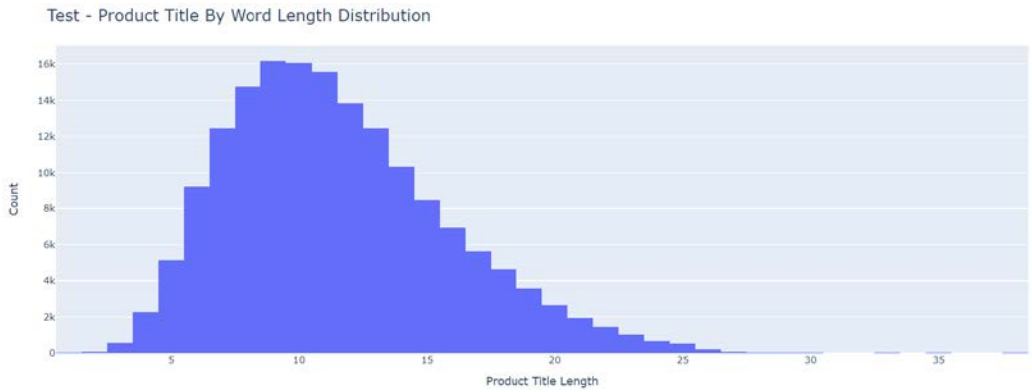
After seeing that the relevance distribution isn't equally spread, we checked that it similarly occurs in the test case as well, therefore, we concluded that the model requires a train and validation split that could be randomly chosen. In our case we took the last 20% of the train set in favor of the validation set.

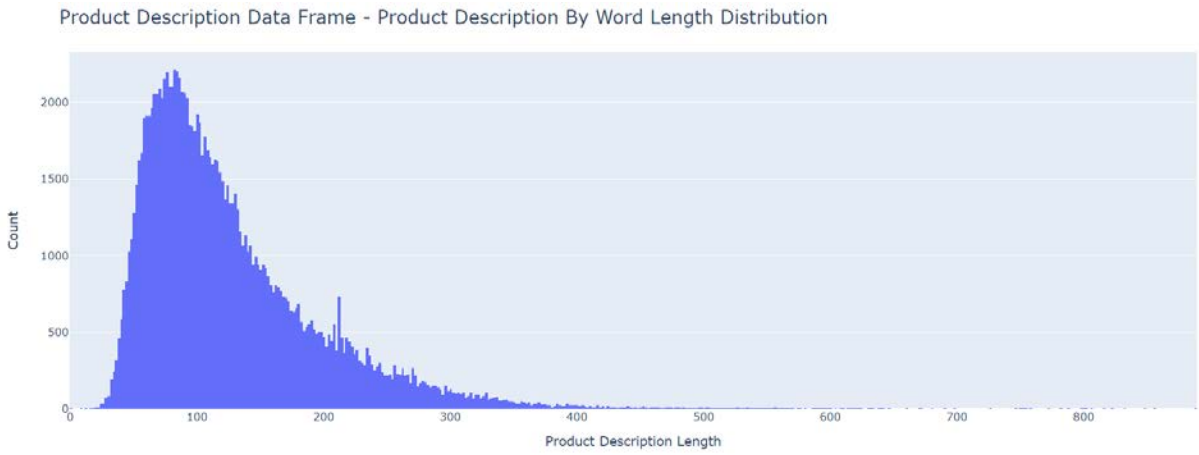




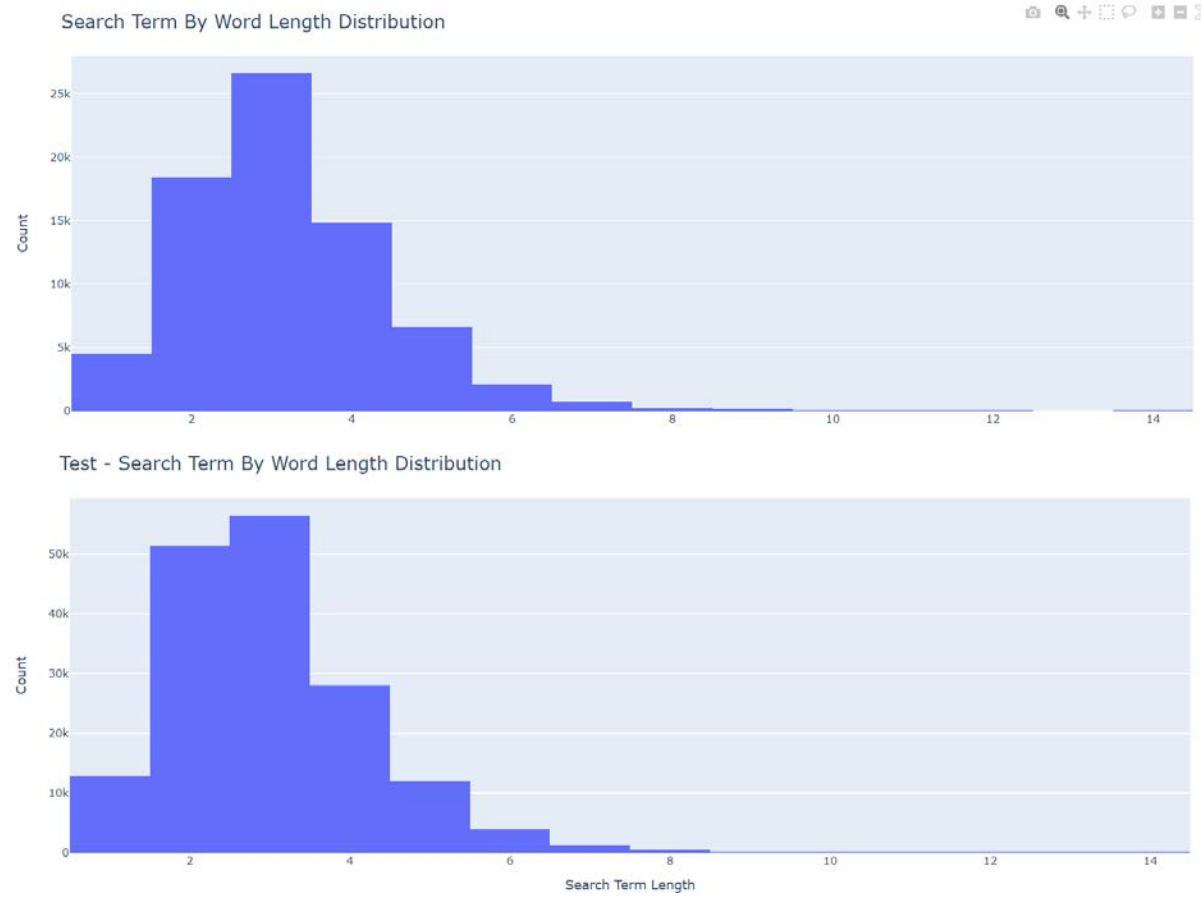
We can see that there's an overwhelming quantity of titles with a length of up to 25 words, despite that, we decided to use the longest product title length when padding the inputs because we wanted to ensure our model receives every single word.



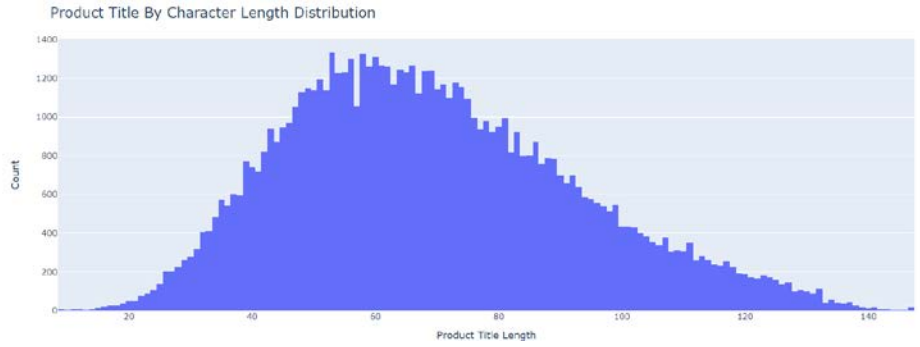


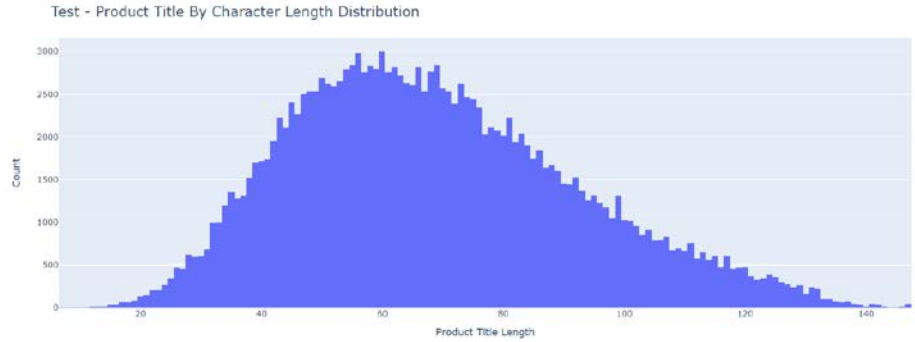


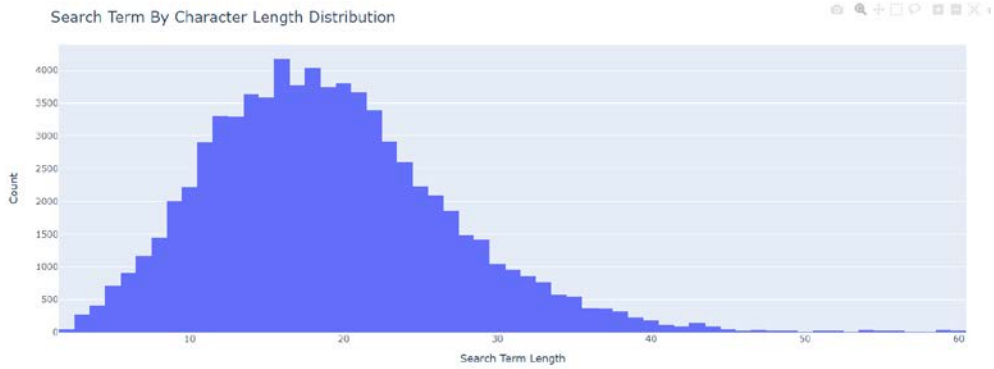
The length of the search terms isn't similar at all to the product title, and even less similar to the product description, which we decided to omit in order to reduce the length of our padding.

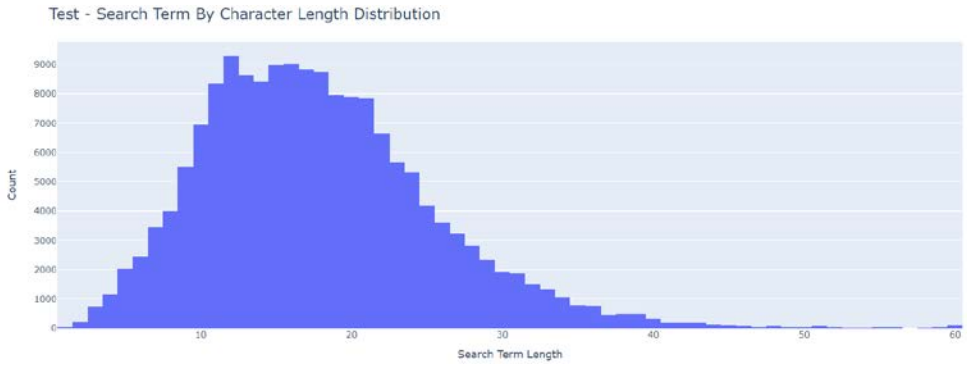


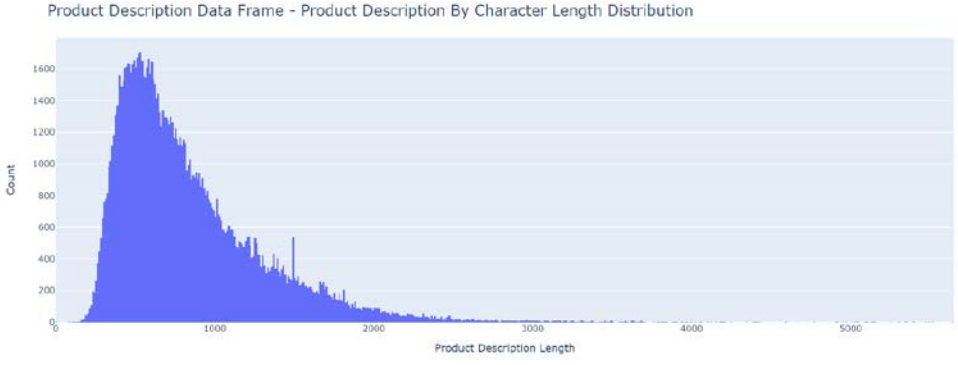
In contrast to the word length, we decided to use a length of 50 for the character level, because the padding in the character level case is affecting the model more acutely.





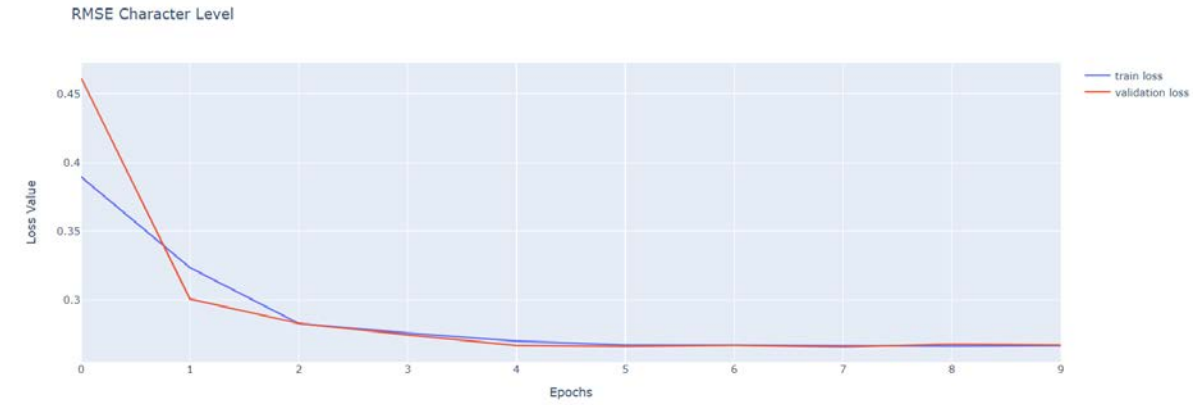


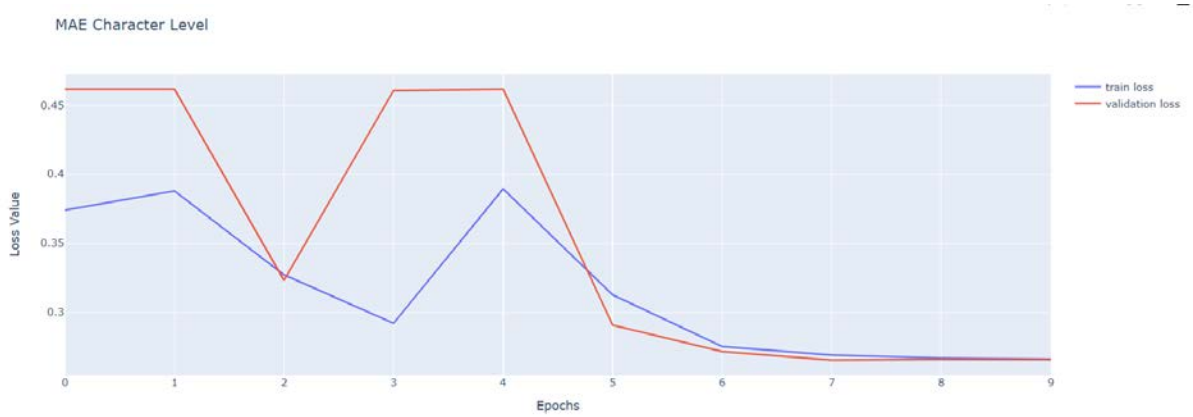




Training Process Plots:

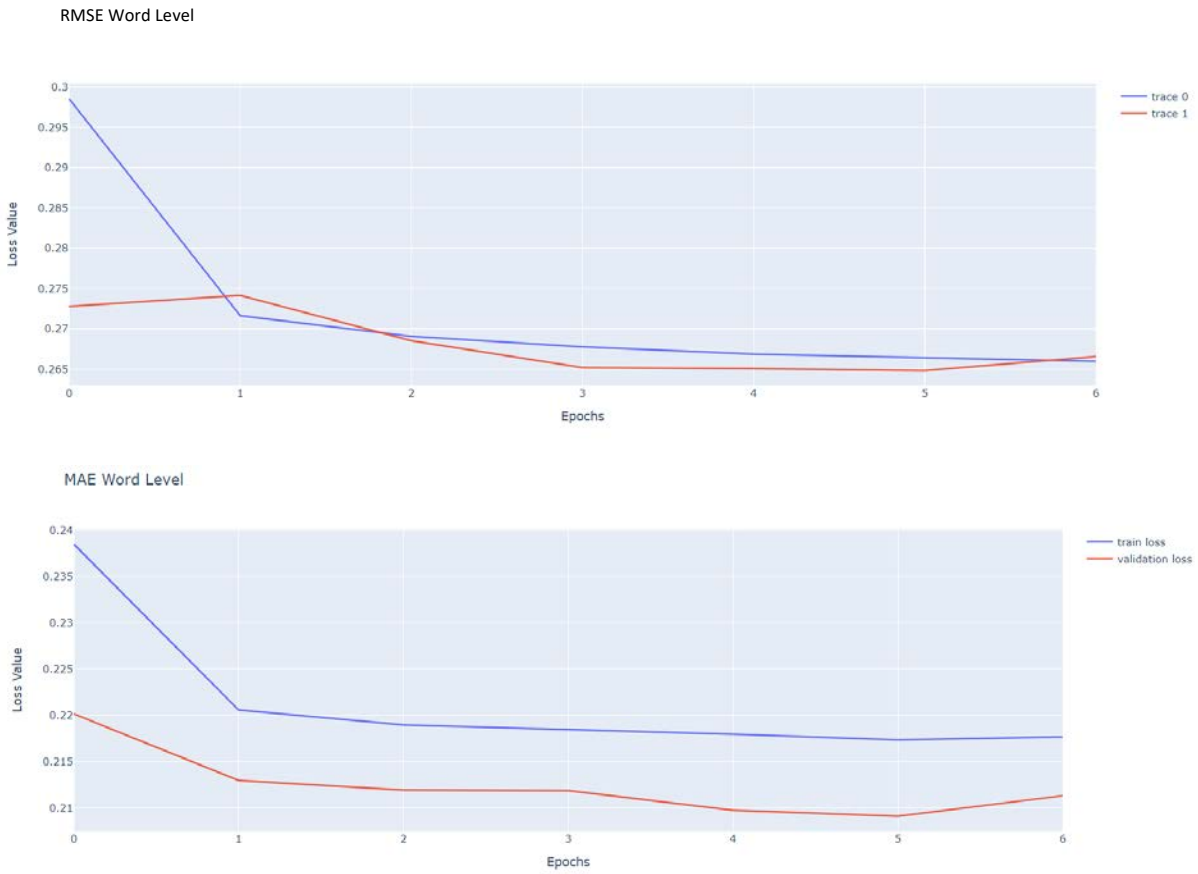
Character Level LSTM:





As we can see from the plots displayed above our model converged into a good fit using the RMSE loss function yet had trouble when attempting to converge with the MAE loss function. We attribute this to the fact that most values are high, and every change is more significant.

Word Level LSTM:



As we can see from the plots displayed above, both loss functions appeared to yield similar results with the MAE variant yielding a little better results.

Results Table:

Model Type	Runtime	Train-RMSE	Val-RMSE	Test-RMSE	Train-MAE	Val-MAE	Test-MAE
Naïve Benchmark Model	48.5 sec.	-	-	0.5921	-	-	0.4750
Character level LSTM	4 min. 34 sec.	0.2668	0.2673	0.2792	0.2662	0.2659	0.2762
Word Level LSTM	2 min. 33 sec.	0.2659	0.2665	0.2795	0.2176	0.2112	0.2661
LSTM Character Level FE for XGBoost	9.87 sec. fit 68.7 ms. predict	-	-	0.5430	-	-	0.4453
LSTM Character Level FE for LightGBM	5.95 sec. fit 78.1 ms. predict	-	-	0.5371	-	-	0.4404
LSTM Word Level FE for XGBoost	2.67 sec. fit 67 ms. predict	-	-	0.5399	-	-	0.4423
LSTM Word Level FE for LightGBM	3.18 sec. fit 111 ms. predict	-	-	0.5367	-	-	0.4399

Final Remarks:

Throughout this project we had an introductory meeting with a new type of architecture - a Siamese Network. This new type of architecture involves both familiar concepts that we have previously explored, as well as various twists on some of those concepts to introduce a new form of data manipulation.

We were introduced throughout the course to various uses of text and their embedding in a common domain, and through this project we were finally able to put it into practice.

We also learned that in the construction of a model of this size we should take into consideration many factors such as the window size, hidden layer amount, appropriate pre-processing of the data.

In this project we were tasked once again with putting various classic machine learning models to use. This solidified the importance of the usage of some well-known algorithms to serve as a stepping stone for the construction of a full-fledged network, both as a pillar and as a benchmark.

To our surprise, we weren't expecting a model that was trained on a character level to be as successful as it had gotten. In contrast to the way humans perceive sentences and language, disregarding the sequence of characters involved, the model was successful in making these connections and distinctions (even after the removal of whitespace characters that would otherwise mark the end of a word or phrase for a human reader). It exposed us to a new way of performing semantic analysis that is not hindered by human perceived limitations.

The world of natural languages itself is an intriguing topic to expand our horizons with further down the road. Common embedding domains that are used to bring various real domains together, similar to the likes of DALL-E, is especially fascinating.