COSC420: Transformer based language model

Jake Norton (5695756)
May 22, 2024

1 Introduction

I have created a framework to generate text both from a combination of Pride and Predjudice and War and Peace as the datasets. Many of the decisions made here were close to arbitrary in nature due to time constraints, and I did not get to explore as much as I would have liked. However, I do believe that the models created are still adequate at creating text that is at least interesting. In terms of text choice, I decide to use all of both books. This provided the most data, but at a cost of potential bias. This was a trade off I was willing to take as I hoped that generally the model would perform better, if skewed towards talking about war and peace

2 Task 1: Tok2Vec Encoder

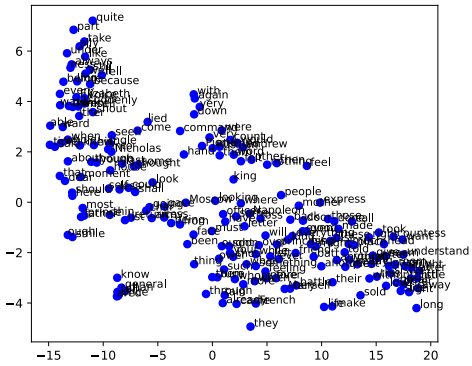
2.1 Methodology

There are two general approaches taken to create word2vec embeddings, Skip-gram(SG) and continuous bag of words(CBOW). They are inverses of each other, CBOW is designed to predict a word given its surrounding context words. SG therefore from a given word tries to predict its context, ie, the words surrounding it. Generally, CBOW works well in contexts where the emphasis is finding common connections between common words. For a small dataset this is useful as it is looking for more simple relationships, and so needs less data to find them. However the averaging which occurs over all of the context can result in a less nuanced view. Skip-gram however tends to pick up more complicated relationships at the cost of needing more time and more data to train, it needs to train separately for each position in the conteaxt window to be effective, whereas CBOW can add all the context at once. I decided to use Skip-gram as I thought it would be more interesting to see if it could still perform ok given the dataset constraints. Part of the reason for the lack of exploration in the process was due to a corruption of my early token-to-vector models. In order to speed up the process I decided to save the encoded vectors of the entire corpus, and simply read it in each time. This did significantly improve the time taken, however somewhere along the line this dataset was corrupted. When looking at the TNE plots of the models, they seem like they are doing somewhat what is expected, creating clusters around semantically similar words. However, as I found out later, when running a prediction on these models, the output would be all the same, some combination of

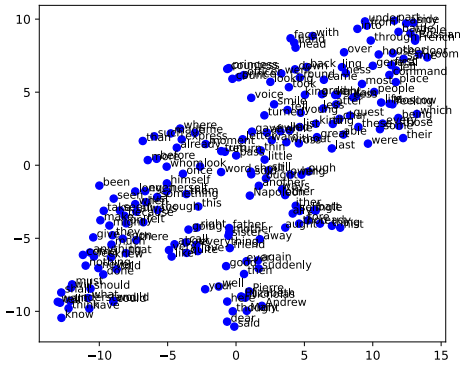
- I
- /ae
- =

This rendered the resulting transformer models to produce garbled output as well. Below is the hyperparameters I used in the end for my token-to-vector model. In terms of context window, anything below 3 didn't give me good outcomes when it

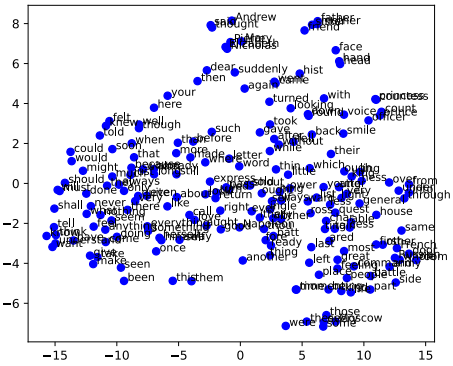
came to the prediction test, I went up to 5, choosing between 3,4 and 5 was difficult, mainly the difference was that the higher the context window, the higher it priorized punctuation. As the decision became somewhat arbitrary I decided to go with 4 in the hope of splitting the difference as well as reduction in training time. I am happy in the end with the split of punctuation vs text, however quotations are still a point that the model often struggles with. Additionally the models with context windows of 4 tended to have a nice evenish distribution, with some notable clustering with expected semantically similar words like "would, should , could, must, shall".



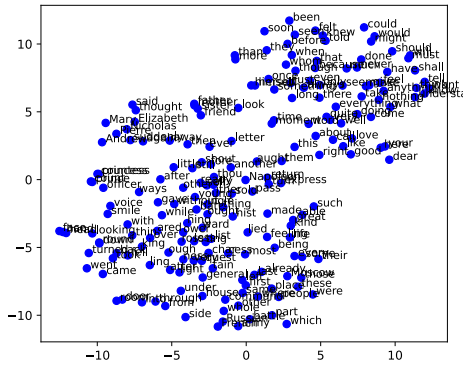
(a) hidden layers: 32, context window of 1



(b) hidden layers: 32, context window of 4



(c) hidden layers: 64, context window of 3



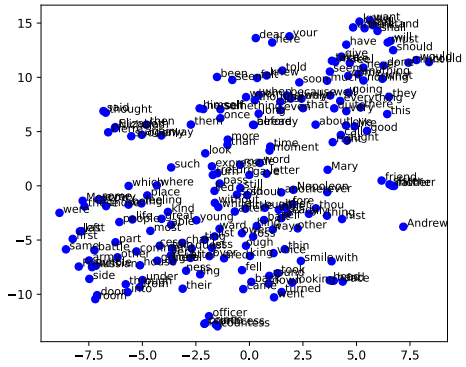
(d) hidden layers: 64, context window of 5

Figure 1: t-SNE plots of embedding space with varying sizes of hidden layers and context windows

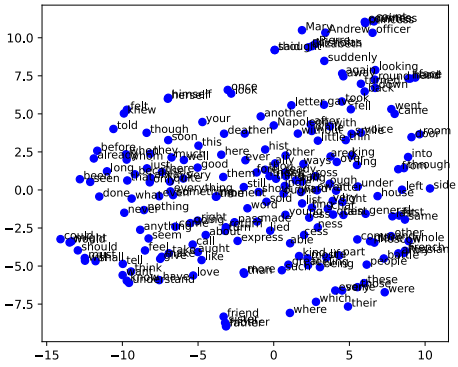
To find out the vocab size I calculated a rough count of unique words[1] over the whole corpus which gave me approximately 25,000 words. From the eye test, though, there were many duplicates/close duplicates in there, mainly due to punctuation, for example.

- 'your
- "your

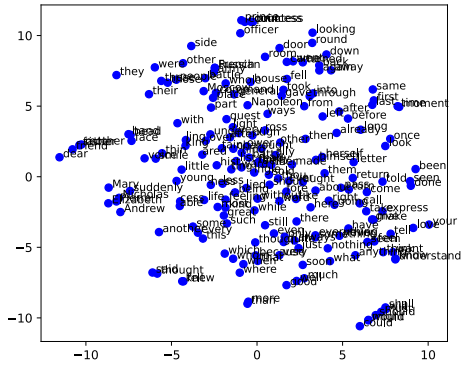
[1]bash command: `tr '[:upper:]' '[:lower:]' < combined_books_text | tr -d '[:punct:]' | tr ' ' '\n' | sort | uniq | wc -l`



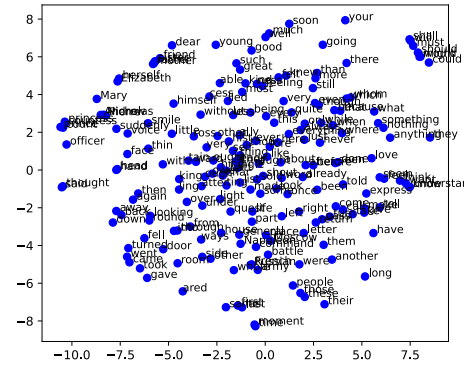
(a) hidden layers: 64, context window: 4



(b) hidden layers: 128, context window: 4



(c) hidden layers: 256, context window: 4



(d) hidden layers: 512, context window: 4

Figure 2: t-SNE plots of embedding space with varying sizes of hidden layers and context windows

- your
- “you’re
- you’re

To capture as much as I could I went up to 5000 tokens which was the limit due to memory and model size concerns at higher token counts. Additionally, if chatgpt can manage with only 50000 unique tokens, 10% for a corpus of this size seems reasonable to me.

I tested a range of values for the the hidden layer dimension, from 32 up to 512. From my findings again with limited testing it was hard to quantitatively decide which would be the best to continue with. For sure I could tell that 32 and 64 were not enough as there was a heavy skew present in the graphs output from all of these dimensions. Due to how t-SNE squishes the dimensionality, I am not sure that having a linear skew is necessarily a bad thing, however my interpretation was that the skewness represented a simpler embedding representation.

I chose to go with the middle of the good values again.

- **Context Window**: 4
- **Hidden Layer Dimension**: 256
- **Vocab Size**: 5000

2.2 Implementation

2.3 Results

Visualization of the vector embeddings using t-SNE and K-means clustering, along with a discussion on the distinctiveness and coverage of the vector space.

3 Task 2: Transformer-based Text Prediction

3.1 Methodology

Description of the transformer model architecture, including the choice of hyperparameters such as sequence size, number of layers, self-attention heads, and neurons in the dense network.

3.2 Implementation

Discussion on the implementation details, training process, and the challenges encountered during the model training.

3.3 Results

Analysis of training and, if applicable, validation accuracies. Qualitative assessment of the text generation capabilities for both the one-hot encoded and Tok2Vec encoded models. Comparison of outputs when prompted with extracts from the training texts versus new, unseen prompts.

4 Task 3: Report and Evaluation

4.1 Overview of Tasks

Summary of the methodologies and key decisions taken during the project.

4.2 Evaluation and Comparison

Critical evaluation of the models based on the tasks' results, discussing the effectiveness and limitations of each approach.

5 Conclusion

Reflections on the project outcomes, lessons learned, and potential areas for future research.

6 References