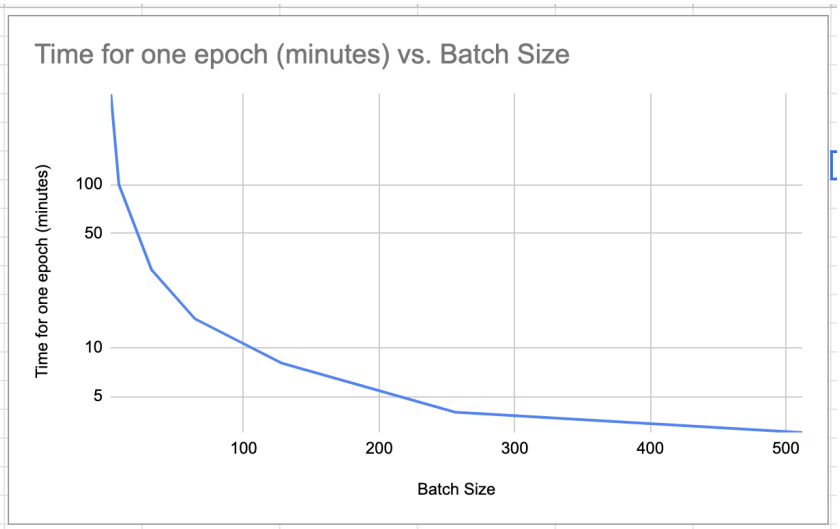
SI 330 HW2 - Word2Vec and Attention Based Classifier

Kaggle Username: https://www.kaggle.com/chittaranjan19

Problem 12: As mentioned on Piazza @127, I used different batch sizes to get a rough time estimate of training time per epoch, and plotted it manually on Excel.

Batch Size	Time for one epoch (minutes)
2	360
8	100
32	30
64	15
128	8
256	4
512	3



There is quite a long “knee” to this plot, so I’m considering batch sizes between 64 and 128 as the optimal value. I started noticing memory issues after batch size 256 so this would not have been ideal on a laptop grade computer.

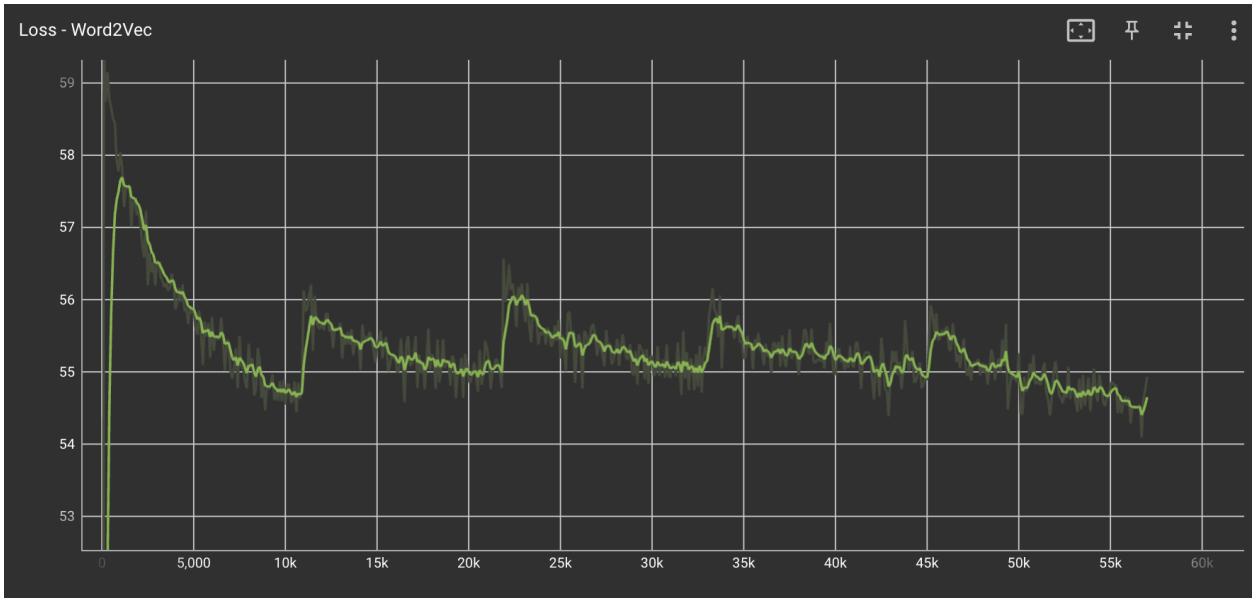
The trend is clear from this plot that as the batch size increases, training time decreases (seemingly exponentially) but plateaus out once the batch size is high enough (~256) because of the overhead of having to swap large amounts of data in and out of memory.

Problem 13: Word2Vec Training Loop

Hyperparameters:

```
EMBEDDING_SIZE = 50
LEARNING_RATE = 5e-4
BATCH_SIZE = 128
EPOCHS = 2
```

I used a higher learning rate than recommended, which I think explains the spikes in the loss values. Overall I think the model still did converge to a reasonable configuration.



Problem 15: The Word2Vec Vector Analysis notebook has example words that I tried to find the similarities for. In general, I noticed that the most similar words were qualitatively coherent for common words (books, good, laptop, etc) but did not make much sense for less common or rare words (yolo, encyclopedia, etc)

This makes sense because the word vectors are largely dependent on the volume of the training data the neural network has seen, as well the type of data. Since the data was product reviews, words related to that domain are the ones that are best learnt and represented.

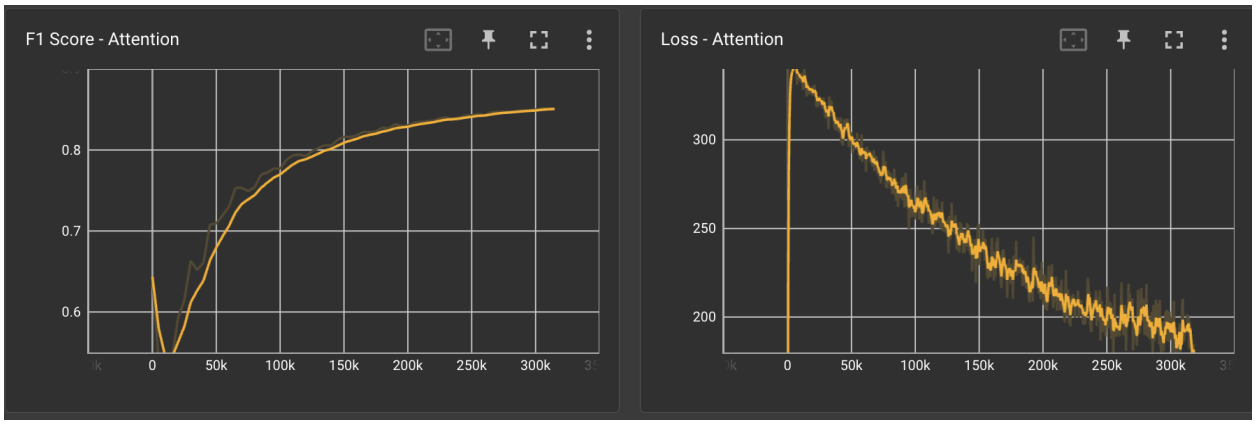
Problem 16: The Word2Vec Vector Analysis notebook has the analogies which are also presented here:

1. `Man - Boy = Woman - Girl`
2. `Book - Chapter + Page = Booklet`
3. `Worst - Bad = Greatest - Good`
4. `Mumbai - Traffic + Clean = Relaxing`
5. `Texas - Shooting + Education = Policy`
6. `Florida - Crocodile + Dog = Washington`

Once I got the model running, this part was fun and I spent way more time than I should have on this. I started off (with no expectations) with the famous King - Man + Woman analogy, and my model's response was "Tom" which was hilariously unfortunate. Once I got to playing around with analogies specific to the reviews domain, I was able to get some good ones like #2 and #3.

Then I realized I could play around with locations and traits about those locations (although I'm not entirely sure how the model was able to learn these embeddings that well from product reviews) which led some interesting analogies like #4, #5, and #6.

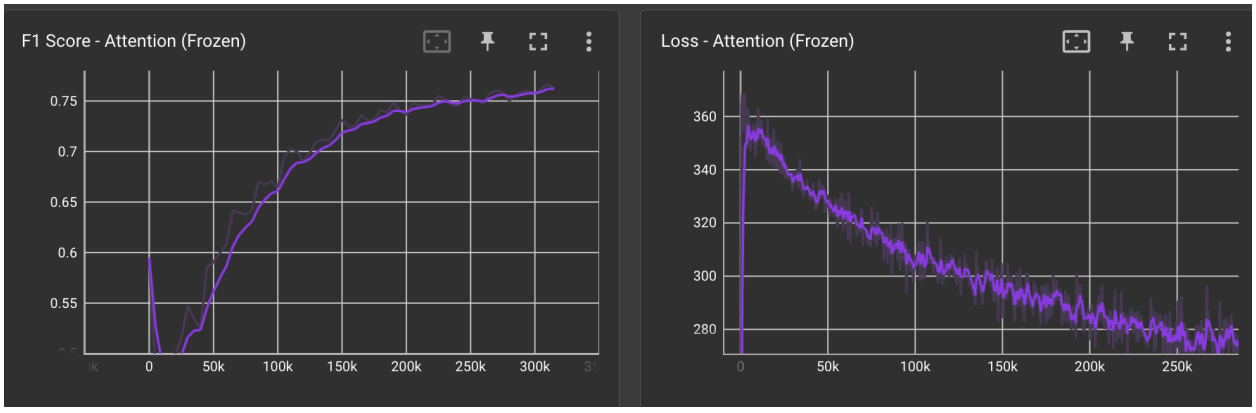
Problem 18: Training Attention Based Classification Model



Problem 19: Frozen Word Embeddings

The model seems to converge just as quickly (if not quicker) and there is a significant boost in training time due to the lower number of parameters. While there is an impact on the F1 score, my guess is that with more training time that will also reach the same score as that of the unfrozen model.

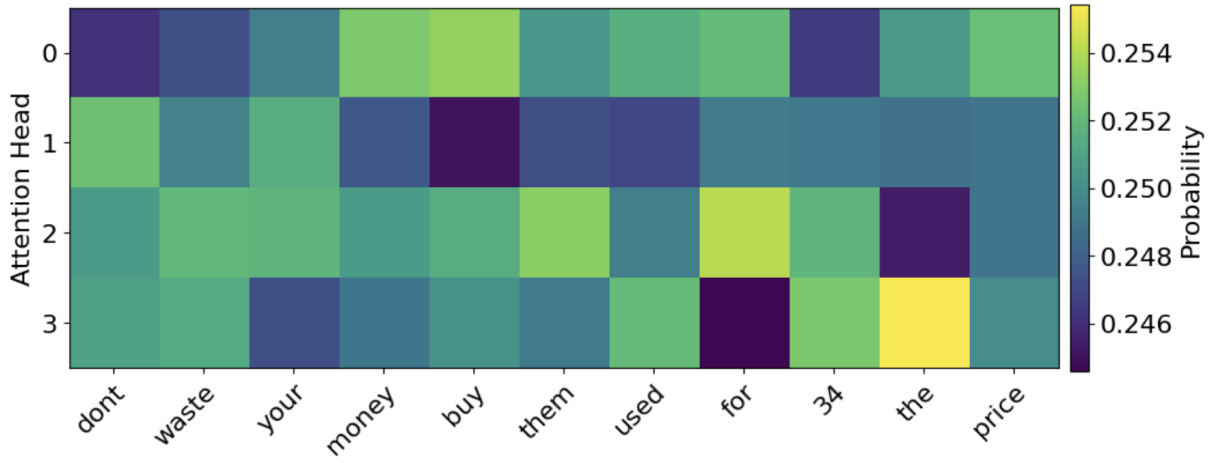
Not retraining the word embeddings make sense, because this neural network is specifically designed for a sentiment classification task, and might not necessarily be representative of a vocabulary using which word embeddings could be trained.



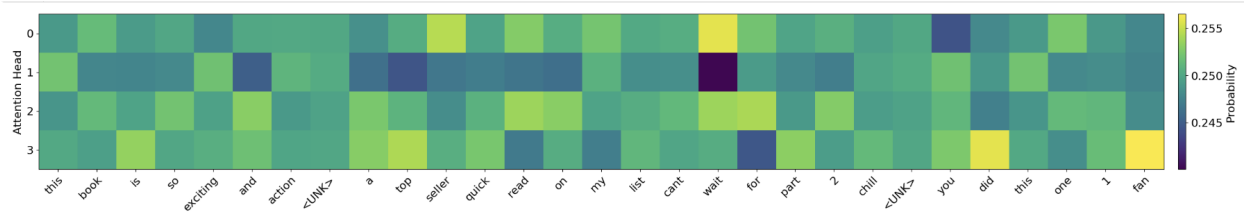
Problem 21: Attention Head Interpretation

I couldn't find a clear interpretation for the attention heads. Their values were all very close to each other for most documents (within 0.01) and I wasn't sure if that difference was worth differentiating with.

But if I had to guess based on most examples, it looked like Head #4 was looking at stop words like "the"

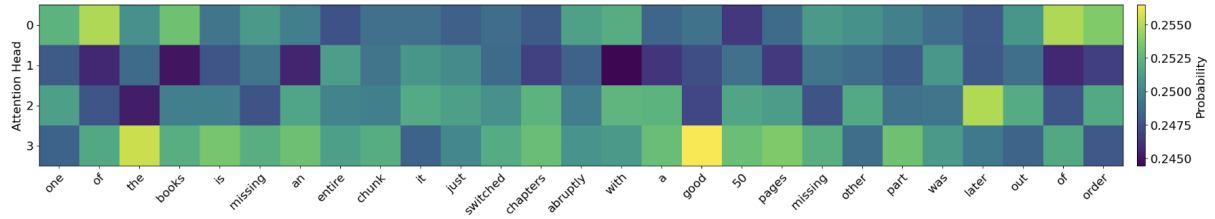


Head #1 and #2 seemed to be looking for opposite things, as shown in this heatmap, for the word "wait"; I'm not quite sure exactly what they are looking for though.

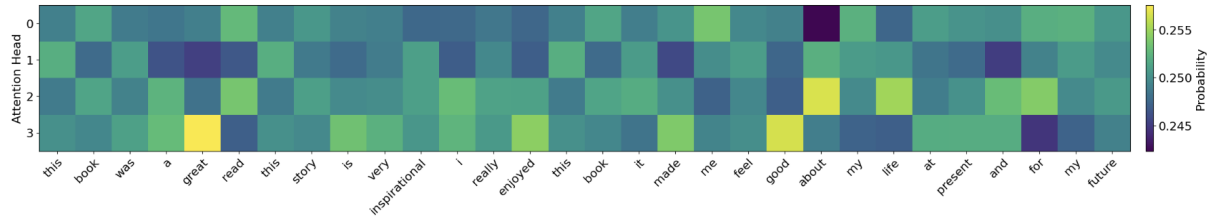


It was tricky to find more meaningful examples, but in the following two it seems like Head #4 is looking for positive words like "good" and "great":

```
for s in sent_dev_df[sent_dev_df["label"] == 0].sample(1)["text"]:
    words = [index_to_word[index] for index in s]
    pred, tokens, attn = get_label_and_weights(" ".join(words))
    visualize_attention(tokens, attn)
```




```
for s in sent_dev_df[sent_dev_df["label"] == 1].sample(1)["text"]:
    words = [index_to_word[index] for index in s]
    pred, tokens, attn = get_label_and_weights(" ".join(words))
    visualize_attention(tokens, attn)
```



Considering that there is no clear idea on what exactly the heads are looking for, I found it tricky to look for a counterexample where the classifier is incorrect and analyze what the heads are looking at.