Lilli Lewis - May 16
Lech Szymanski - COSC420

Assignment 2

Task 1:

For Task 1, I loaded the text using the provided load method, set the vocab size, created a tokenizer, trained the tokenizer on the text, and encoded the tokens using the provided 'encode' method. I then prepared the data to be used in my model by selecting a window size of 2, making empty input and output arrays, and looping through the entire text and putting the correct words in the correct arrays (the center word went in the input array however many times it had a corresponding output word, and each surrounding word within the window size went into the output array). I switched the input array to one-hot encoding, converted the output array into numpy format, and then the data was ready! I built a sequential model with a single hidden dense layer with 768 neurons (Lech recommended 768 neurons). I used a softmax activation function. My output layer has the same number of neurons as vocab size, a softmax activation, and zero bias (also recommended by Lech). I compiled the model using adam and categorical cross entropy. I fit the model with the input array, the numpy version of the output array, and the same number of classes as the vocab size. I used a batch size of 32. I saved the first layer of weights from the model, and plotted the results of that first layer of weights with the word index from the tokenizer, plotting 100 words.

I've tried a few combinations of vocab size, which texts to use, and number of epochs. I found that as I upped the epochs, the loss decreased, which I know is important. Originally, I wanted to use both texts and a vocab size of 1000, but I got an error that the computer was unable to allocate 21.5 gigabytes to run the model. So, I switched to using only 500 vocab words, but each epoch took 10 minutes, and knowing that I wanted to use a higher number of epochs (like 40, to decrease loss), I decided that was against my best interest. I switched to using only *Pride and Prejudice* (henceforth P&P), with a vocab size of 500, and got the following tok2vec plot:

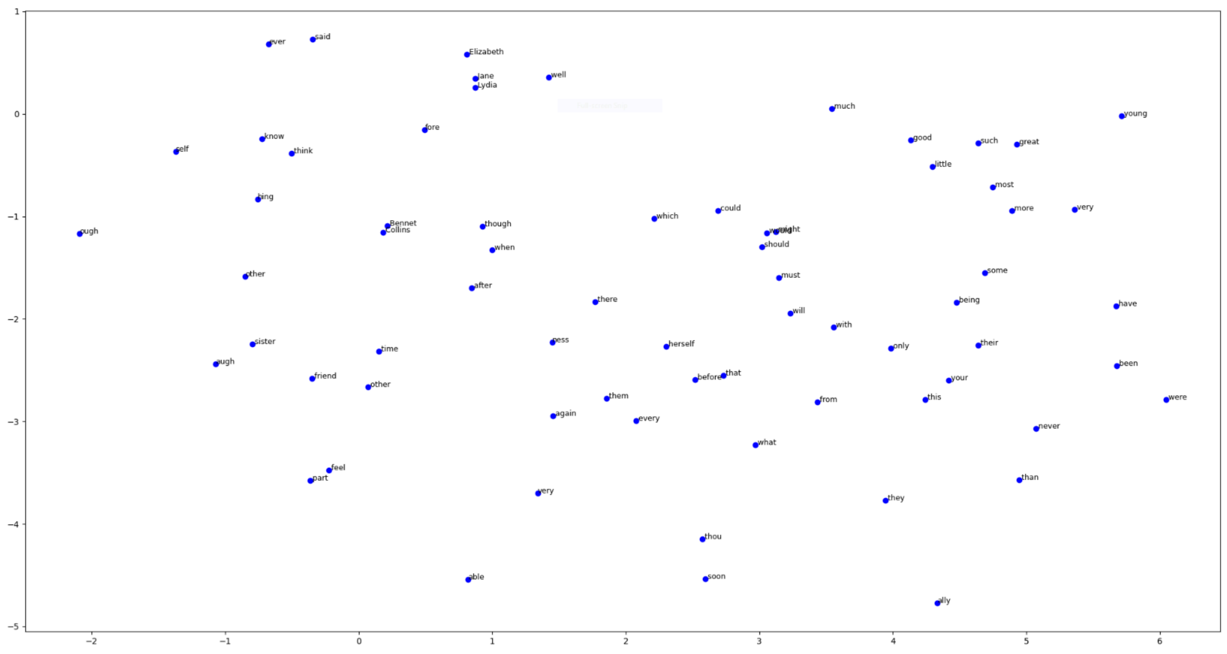

***Figure 1**: Model 1's plot_tok2vec*

It seems to have worked reasonably well. It grouped Elizabeth, Jane, and Lydia together, who are, to my understanding (it's been like 8 years since I read the book, but I did look it up), sisters. It also placed "sister" and "friend" close together. It placed "most", "more", "very", and "great" together, which all describe an exaggeration or greater amount. It placed "good" near "great", which makes sense. It puts "Bennet" and "Collins" together, which is, I think, because they were likely mentioned close to each other (all I really remember is that they're both men in the book). It placed "know" and "think" together, too. For the groupings I described, I think this was reasonably successful. The loss on the final epoch was 4.7506, which I know is high.

I also ran the code with P&P, a vocab size of 1000, and 10 epochs, which takes about 3 minutes per epoch. The loss at the end was 5.2655, which I know is high, and I think more epochs would lower it since it was lowering with each epoch. However, I'm not willing to wait longer than 30 minutes for each run.

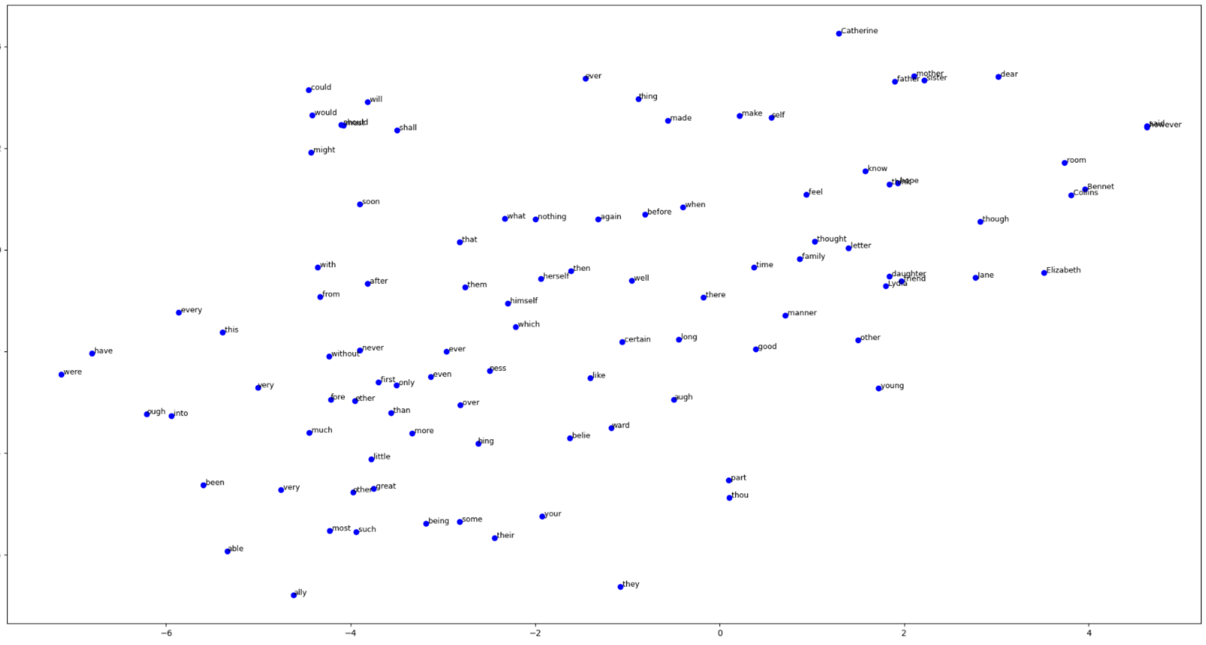

***Figure 2**: Model 2's plot_tok2vec*

This appears successful, too. "Mother", "father", and "sister" are all plotted extremely close together. "Bennet" and "Collins" are together, too. "Know", "hope", "think", and "feel" are together, which all makes sense, as are "thought", "family", "letter", and "time". Given the context of the book, where I seem to remember the protagonist writing letters to her family members pretty frequently, this makes sense too. "Most", "much", "more", and "great", are in the same part of the chart, but less close than before. "Himself" and "herself" are pretty close. "Could", "would", "will", "should", "shall", and "might" are all super close together, which is good because they are semantically very similar. "Daughter", "friend", and "Lydia", are extremely close together, which I think makes sense in the context of the book. "First", "only", and "ever" are close, and they're near the "more" grouping, which I think linguistically also makes sense. For these reasons, I think this model makes sense.

It is worth noting that I've kept the window size at 2 for both of these. I think I would get much better results with a larger window size, since that would give the model more context for its associations. I'm running the model with P&P, 1000 vocab words, and 10 epochs with a window size of 5 just to see how it works. It’s taking significantly longer, about 10 minutes per epoch, so I’m only going to try this once. I picked the specifications from the second model since I like its groupings better than the first model’s. I’d usually spend more time refining, but I know Lech doesn’t care how well the model works, so I’m not going to refine it more than this and I’ll decide which model is best between the second one and the third (with the only difference between the two being the window size (2:2 and 3:5)).

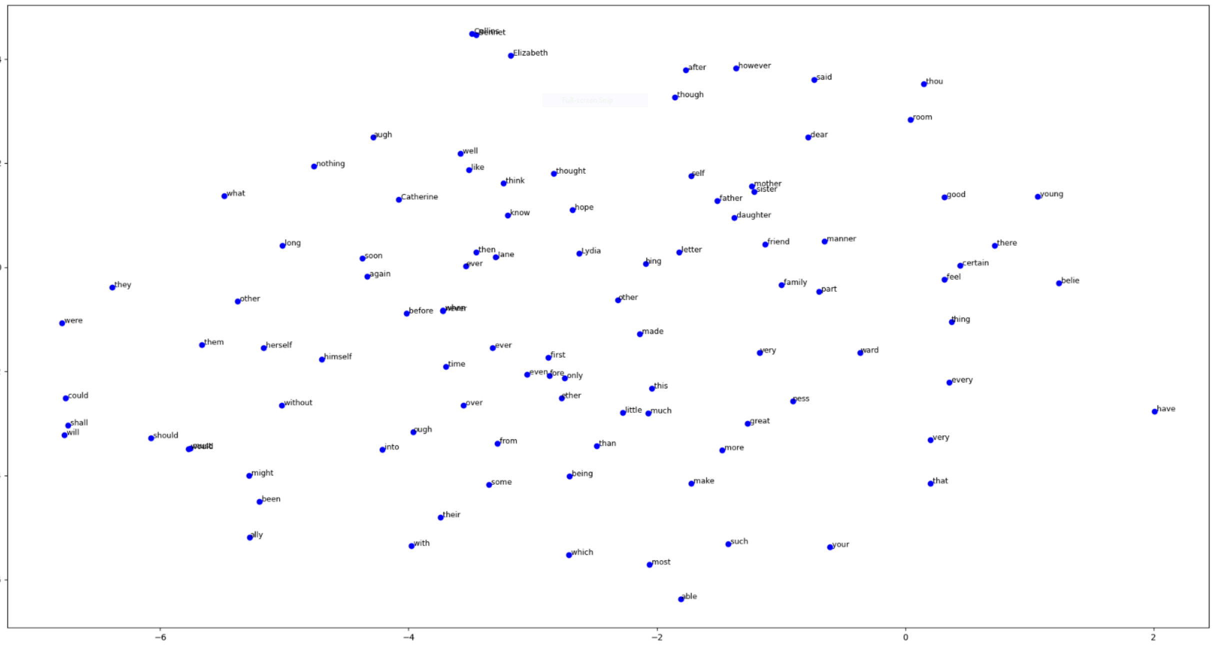

***Figure 3**: Model 3’s plot_tok2vec*

This third model appears to be better than the second. It groups “will”, “shall”, “should”, and “could” together, “mother”, “sister”, “father”, and “family” together, “think”, “thought”, “hope”, and “know” together, “first”, “ever”, “other”, “only”, and “time”, together, and “himself”, “herself”, and “them” together. “Bennett” and “Collins” are together again. The sisters names are more scattered. There is a grouping for exaggerations--”great”, “more”, “little”, “much”, and “than”-- but it’s more spread out. “When”, “again”, and “before” are also together, which all relate to time. I think this is my best model, so I’m sticking with it!

I had a talk with Lech where I mentioned that I had a zero bias setting in the output dense layer instead of the hidden dense layer. He recommended I switch the zero bias to the hidden layer instead of the output layer, so I’m switching the zero bias to be in the hidden layer, rerunning the model, and will report the results.

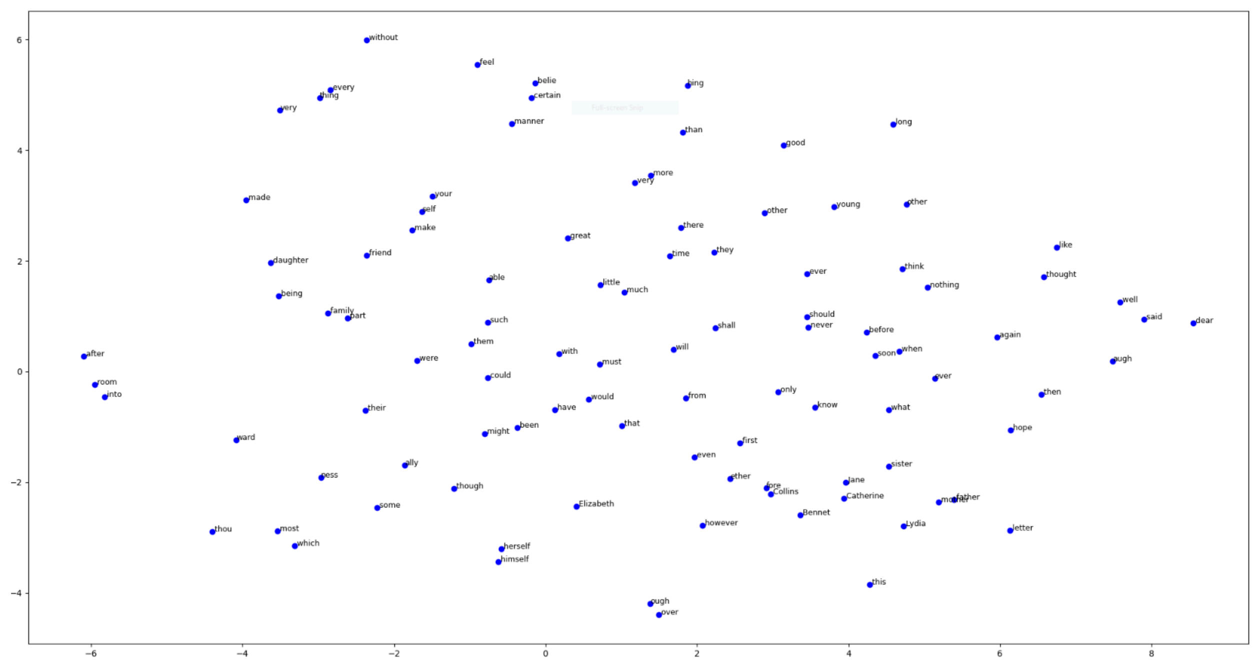

***Figure 4**: Model 4's plot_tok2vec*

The Bennett characters' names are all cluttered together, alongside "sister", "mother", "father", and "letter". "Daughter", "family", and "friend" are all on the other side of the graph. "Himself" and "herself" are close together. "Your" and "self" are close together. "Before", "soon", "when", and "never" are all clustered together, all of which have to do with time, so clearly something is working. To be honest, though, I think the third model is better. The difference between the two models is that the third model has zero bias in the output layer and this fourth model has zero bias in the hidden layer. Lech recommended that I use zero bias in the hidden layer instead of the output layer, though, so I will be sticking with this fourth model. After some research, I think the model might work even better with zero bias in both layers, but I know that accuracy is not the point of this assignment, so I'm leaving it as is with zero bias in the hidden layer instead of the output layer.

Task 2:

I started by taking Lech's example code from the main function of the transformer file. I adjusted it to use my embedding layer instead of the BERT tokenizer. I then had a meeting with Lech and a few other students where he decided we should use his predictTextDataGenerator class because of a padding and masking situation that made his generator better suited to the task than one we would have written.

My model proceeds as follows: if I'm using one hot embedding, the first layer is OneHotEmbedding. Otherwise, the first layer is FixedEmbedding. I then move to a positional encoding layer, then four transformer layers, then one fully connected dense output layer with the same number of neurons as vocab size. I use the provided/taken from tensorflow custom learning rate schedule with the same values used in the example. I found that when I was using one of each layer, my generator would start a loop after around five words, if not immediately, both when using my own embeddings and when using one-hot embeddings. For example, one loop it got stuck in was "It is a truth universally acknowledged that a woman has four or four or four…" repeating "or four" on and on. Another loop it got stuck in was just repeating the word "fortune" over and over. However, after I upped the number of transformer layers to 3 it took much longer to loop, and once I upped it to four it did not loop anymore. I increased the number of heads to 12 because I figured that would increase the quality of the generated text as increasing that value allows the model to pay more attention to more relationships in the data. I also increased the seq_len parameter to be 15, so the model can have longer sentences to work with and hopefully capture some more complicated relationships. I experimented with increasing the number of epochs but found that loss increased a lot and accuracy decreased a lot after epoch 3, so I'm going to use three epochs. Making these changes resulted in an almost exact replica of the words following the prompt, so I tried increasing the key dim to 64, which gave me a different response that also made sense: *"It is a truth universally acknowledged, that a single man of large fortune; four or five thousand a year. What a fine thing for our girls! " " You have no compassion on my poor nerves. " " You mistake me, my dear. I have a high respect for your ner".* I tried increasing the length of the created text from 100 to 200, and got this output: *"It is a truth universally acknowledged, that the experience of three - and twenty years had been insuffffffffficient to make his wife, impatiently. " You want to know who has taken it? " cried his wife, impatiently. " You want to tell me, and I have no objection to hearing it. " This was invitation enough. " Why, my dear, you must know, Mrs. Long says that Netherfield is taken by a young man of large fortune from the north of England; that he came down on Monday in a chaise and four to see the place, and was so much delighted with".* Increasing the length of the output confirms to me that the model is not getting stuck in a loop, which is one of my goals!

This model has a seq_len of 15, vec_dim of 768, key_dim of 64, num_heads of 12, and dff of 256. It's trained only on P&P, runs for 3 epochs, and has one embedding layer, one encoding layer, four transformer layers, and one fully connected dense layer. It uses a custom learning rate. I am now going to run it six times, twice with one hot and twice with the task 1 embeddings. I will run them with two different prompts: one from the text and one that I made up with a plausible line that is not in the text but could be. I'm still only using the first 1000 words from the text since if I use the full text in task 2, it takes 11 minutes per epoch to run.

Table 1: *Accuracy, Loss, and Output Comparisons Between One-Hot and Task-1 Embeddings on A Truth Universally Acknowledged*

	One-Hot	**Task 1 Embeddings**
Masked Accuracy (at epoch 3)	0.8825	0.8598
Loss (at epoch 3)	0.5055	0.5767
Output from "It is a truth universally acknowledged**"**	It is a truth universally acknowledged, that a single man of large fortune from the north of England; that he came down on Monday in a chaise and four to see the place, and was so much	It is a truth universally acknowledged, that the experience of three - and twenty years had been insuffffffffficient to make his wife, impatiently. " You

	delighted with it that he agreed with Mr. Morant like other girls; but Lizzy has something more of quickness than her sisters. " " Mr. Bennet, how can you be so tiresome as Jane, nor half so good - humoured as Lydia. But you are always giving her the preference. " " They are my old friends. I have heard you mention them with consideration these twenty years at least. " " Ah you do not know what I suffer. " " But I hope you will get over	want to know who has taken it? " cried his wife, impatiently. " You want to tell me, and I have no objection to hearing it. " This was invitation enough. " Why, my dear, you must know, Mrs. Long says that Netherfield is taken by a young man of large fortune from the north of England; that he came down on Monday in a chaise and four to see the place, and was so much delighted with

The one-hot-embedding model continued to describe the man in question, then described his action, then described Elizabeth--"Lizzy"-- and her sisters. After some searching, I've found that these are scraps of text from the book that have been pasted together. Mr Morant does not exist, but Mr. Morris does, and his name was sliced in half and combined with the end of "ignorant" and the words following it follow the word 'ignorant' in another part of the text. The text produced is spliced from the prompt: "It is a truth universally acknowledged that a single man of large fortune--", and excerpts: "--a young man of large fortune from the north of England; that he came down on Monday in a chaise and four to see the place, and was so much delighted with it that he agreed with Mr. Morris--", "--ignorant like other girls; but Lizzy has something more of quickness than her sisters." "Mr. Bennet, how can you--", "--"My dear Mr. Bennet," replied his wife, "how can you be so tiresome?--", "--so handsome as Jane, nor half so good-humoured as Lydia. But you are always giving _her_ the preference. They--" etc. Most of these lines come from around line 790, but a few come from around line 750, and of course the prompt comes from 702. The one-hot-embedding model shows that it knows how to put together quotes using the same word in different context, which is a reasonable measure of success for a model with so little training data.

The task-1-embeddings model shows that it knows how to perform the same transitions. For example, the prompt melds with the second quote on the word "that" and the second happens on "his wife". Since the one-hot-embedding model includes splices from 6 quotes, and the task-one-embeddings model only includes splices from 3, this indicates, to me, a little bit more originality/creativity from the one-hot model. The one-hot model was not entirely incoherent, and at the beginning of the prompt, it tries to generate what it knows is the second half of the prompt: "a single man in possession of a good fortune". It generates "a single man of large fortune", which is close, but then it follows that quote and hops along following other quotes. The task-one-embeddings model does not follow the prompt except for the word "that", which is used over 1,000 times in the book, so that doesn't tell me much about how much the model understands the prompt.

Table 2: *Accuracy, Loss, and Output Comparisons Between One-Hot and Task-1-Embeddings on A Doomed Love*

	One-Hot	**Task 1 Embeddings**
Masked Accuracy (at epoch 3)	0.8864	0.8587
Loss (at epoch 3)	0.4848	0.5830
Output from "She loved him but"	She loved him but I do not pretend to be anything extraordinary now. When a woman has five grown - up daughters daughters of she ought to give over thinking of her own beauty. " " In such cases, a woman has not often much beauty to think of. " " But, my dear, you must indeed go and see Mr. Bingley when he comes. " " Is he married or single? " " Oh, single, my dear, to visit him, if you do not. " " You are over scrupulous, surely. I dare say Mr. Bingley will be very glad to see you; and I will send a few lines by you to assure	She loved him but Lizzy. " " I desire you will do any such thing. Lizzy is not a bit better than the other of the other of the others: and I am sure she is not half so handsome as Jane, nor half so good word for my little Lizzy. " " I desire you will do no such thing. Lizzy is not a bit better than the other of the other of the others: and I am sure she is not half so handsome as Jane, nor half so good word for my little Lizzy. " " I desire you will do no such thing. Lizzy is not a bit better than the other of the other of the others: and I am sure she is not half so handsome as Jane, nor half so good word for my little Lizzy. " " I desire you will do no such thing

The one-hot-embeddings model actually performs well on this prompt. The prompt indicates a desire and a lack of success. The one-hot's generated text right after the prompt describes a woman who was giving up on thinking of her own beauty, which is similar to the prompt's description of a woman giving up on love. The generated text goes on to include a conversation between the woman and (assumedly) one of her daughters wherein she convinces her to meet with a single man. This is all related to the prompt.

The task-one-embeddings model performs okay on this prompt: it describes Lizzy and how she's not as good as her sisters. This could be an interpretation of the desire and lack of success in the prompt, especially with the words "I desire you" being in the output text and then describing Lizzy's faults. It does get stuck in a loop, which isn't great, but it appears as though the model understands the prompt.

In my opinion, the one-hot embeddings make the model perform better than the task-one embeddings. It shows more of an understanding of the prompt and an understanding of what it should respond to the prompt. Though both types of embeddings make the model grab and regurgitate excerpts from the text, at least the one-hot embedding regurgitates shorter excerpts and switches around between them instead of picking a scene and replicating it entirely. One-hot embedding model runs have also gotten stuck in loops significantly less than their task-one counterparts.

The loss is pretty high for the model using both embeddings but in both instances the loss is about 10% less when one-hot embedding is used, indicating that the model matches the target values more closely using one-hot embedding than using the task-one embeddings.

The masked accuracy is pretty high: around 85% for the pre-trained embeddings and around 88% for the one-hot embeddings. This also indicates that the one-hot embedding works better than the pre-trained embeddings.

In the future, I would try downsizing all of the numbers (numheads, vecdim, keydim, dff), increasing seqlen, and also try increasing or decreasing any combo of these. I would try increasing the number of transformer layers, also.

I would like to mention that I added all of the model.save code after I did this analysis, so the models and embeddings that you will have if/when you run this code will be a little different than what I had.