Recipe Classifier Report

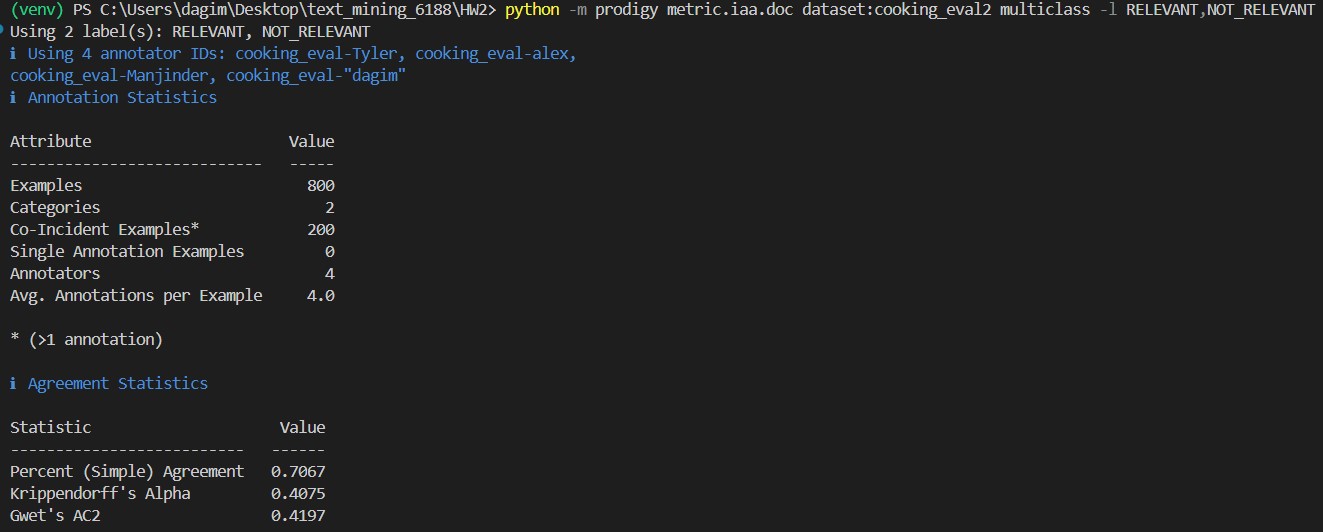
My goal is to create a model that can classify text from a cooking forum on Reddit as either relevant or not relevant. What makes text relevant or not relevant in this scenario is based on the perspective of someone who works for an online recipe publisher who wants to know what recipes are popular or not. Since a lot of the text data from this forum is incredibly useful to actual recipes, it needs to be separated from text that is irrelevant. Creating a text classification model will allow us to filter out the relevant text that can be used by the online recipe publisher.

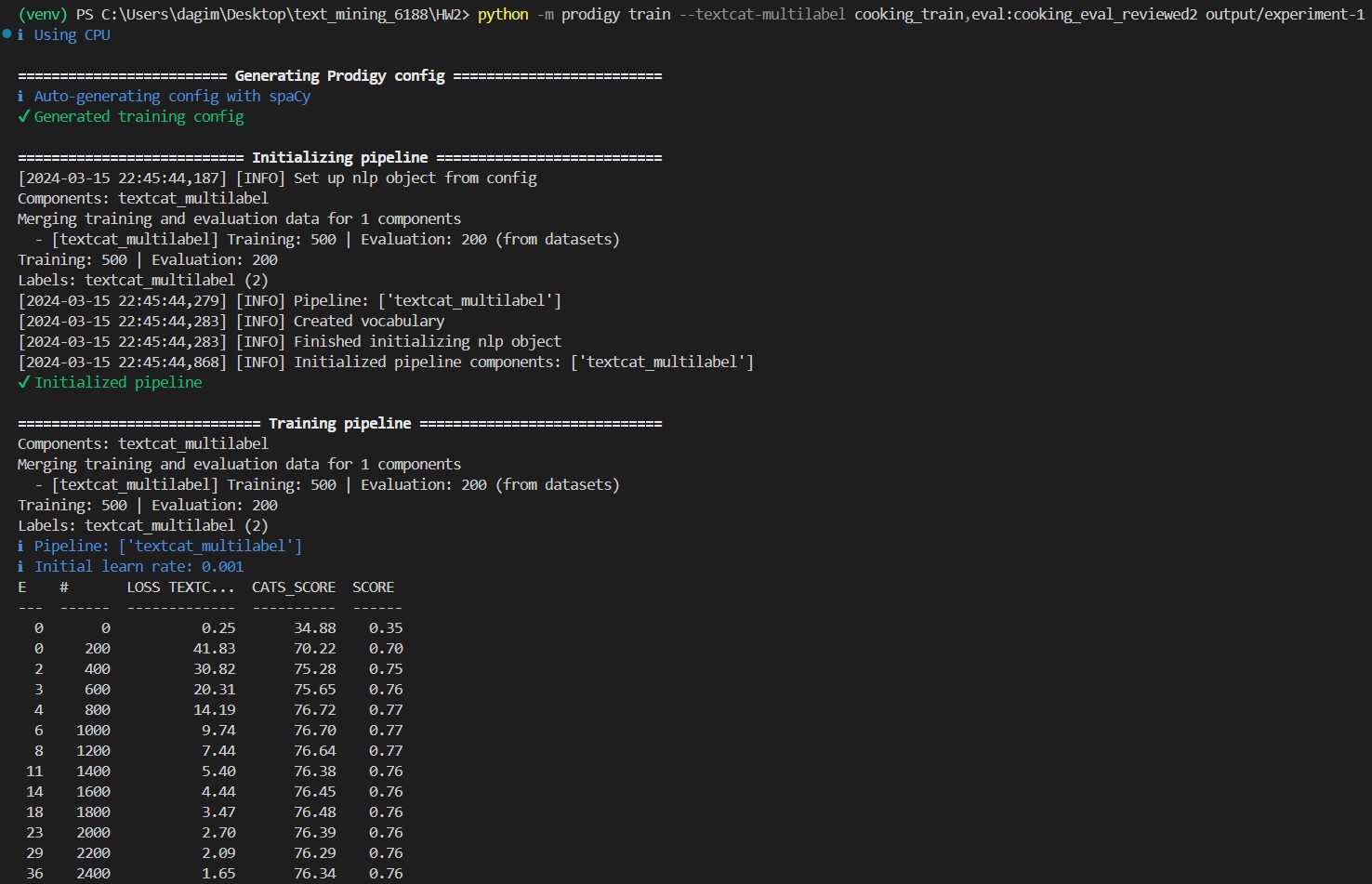
I had access to 5200 lines of text data (5000 of them belonging to the training set and 200 belonging to the evaluation set) with none of them labeled as either ‘RELEVANT’ or ‘NOT\_RELEVANT’. To create the model, I took a semi-supervised approach to training it; this is done by manually labeling a portion of the data so that the model can then learn off of that information. I wanted a “golden” evaluation set to evaluate my models off of so I worked with a group of 3 other people and we each manually labeled the 200 entries of the evaluation set. To do this, I started a local server through prodigy to label the text using the code: ‘*python -m prodigy textcat.manual cooking\_eval homework2\_evaluation.jsonl --label RELEVANT,NOT\_RELEVANT –exclusive’*. This allowed all four of us to provide our own insight into the labeling of the data using session names and Ngrok so that our work can then be shared among us. It is not enough to just combine all our work to create a dataset of 800 entries, we must consolidate our manual labeling efforts into a dataset with only 200 entries, the way it was originally, using the code: ‘*python -m prodigy review cooking\_eval\_reviewed cooking\_eval --label RELEVANT,NOT\_RELEVANT --auto-accept’*. To then determine the consensus of our group of four with how we labeled the evaluation set, I created an inter-annotator agreement (screenshot provided in appendix) using the code: ‘*python -m prodigy metric.iaa.doc dataset:cooking\_eval\_reviewed multiclass -l RELEVANT,NOT\_RELEVANT*’. The output for the inter-annotator agreement has a percent (simple) agreement of 71% and Krippendorff’s alpha of 41%. The Krippendorff’s alpha might seem a little low, but when reviewing the dataset, it was clear that all 4 of us agreed on most of the labels, and even where we were not unanimous, three of the four agreed anyway.

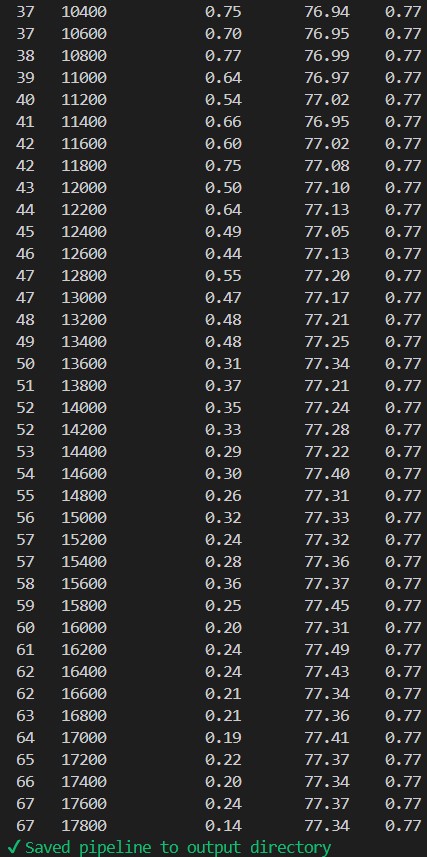
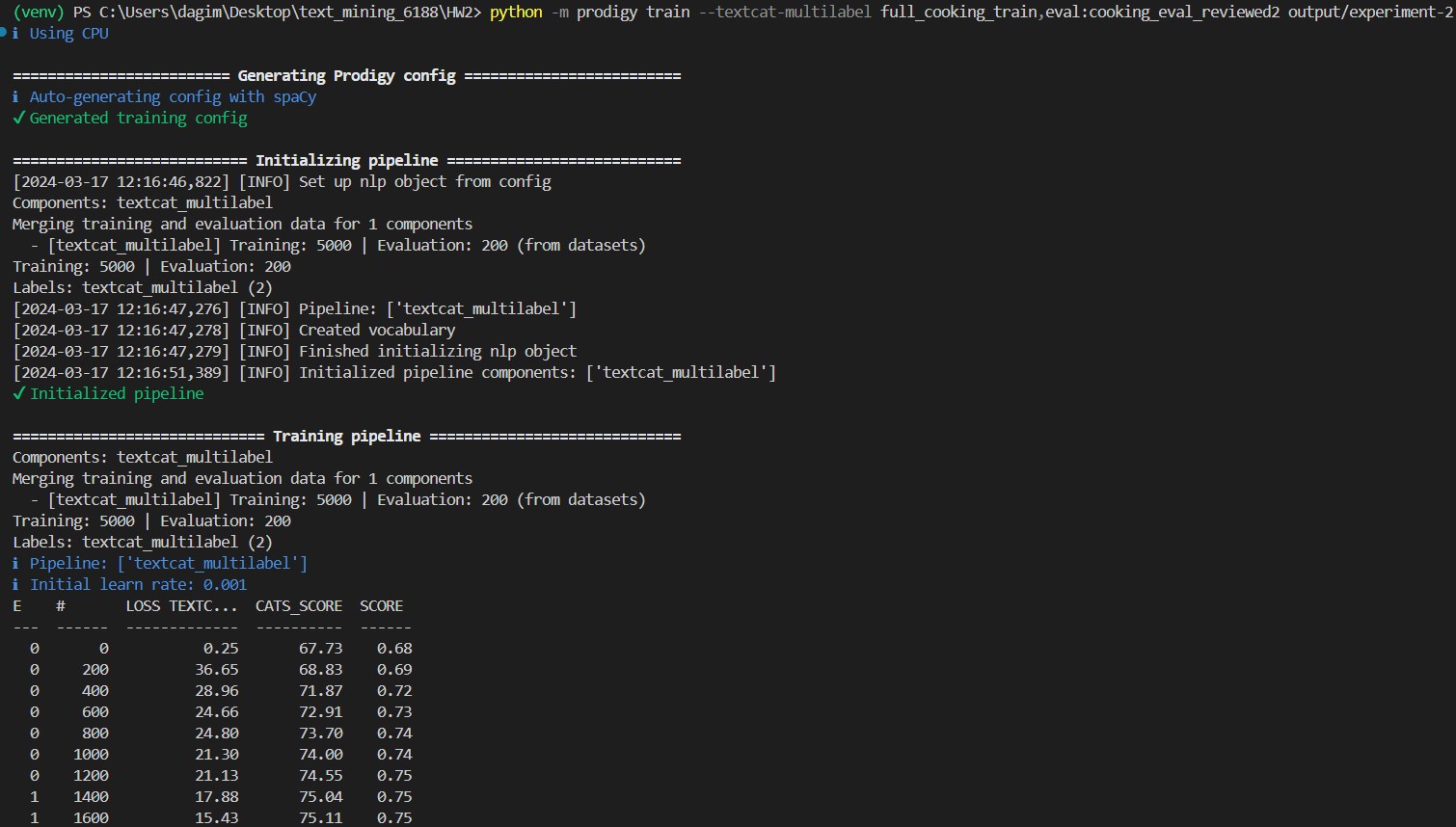
I decided that the best way to find a good text classification model is to conduct four different models, each trained in a different way. I first had to prepare the training dataset for the model. Out of the 5000 lines of text data in it, I decided to manually label 500 of them so that I can then weakly label the rest of the 5000 to then use in the model. For experiment one though, I just used the first 500 as the training set that I called cooking\_train. The highest accuracy I was able to get from this model is 0.77 (screenshot provided in appendix). To conduct the next experiment, I want to use the entire 5000 long training set, but there was a problem formatting the other 4500 lines so that it could combine with the manually labeled 500. I asked ChatGPT to provide me code that can format the dataset. Using a spaCy model and the code provided by ChatGPT (prompt provided in screenshot), I was able to get a correctly formatted complete dataset which I then ran for experiment two. The model for experiment 2 with the full 5000 lines of text training dataset also gave me an accuracy of 0.77. The next experiment used data-to-spacy to use spaCy train with a base model of ‘en\_core\_web\_lg’ along with the complete 5000 long training dataset which provided a maximum accuracy of 0.83 (screenshot provided in appendix). The last model purely uses a hugging face model, specifically ‘distilbert\_base\_uncased’, where I got an accuracy of 0.765 (screenshot provided in screenshot).

Based on the accuracies of the four different models, the best one is model three where we used the ‘en\_core\_web\_lg’ base model with the full training set that was labeled weakly and trained with ‘data-to-spacy’ and ‘hf.train.textcat’. Experiment 2, despite having a much larger dataset than experiment 1, was exactly as accurate as its smaller predecessor; meaning all the extra lines did not help the model accuracy. The hugging face model was the least tuned to the dataset (not being trained on my manual labeling) and thus had the smallest accuracy.

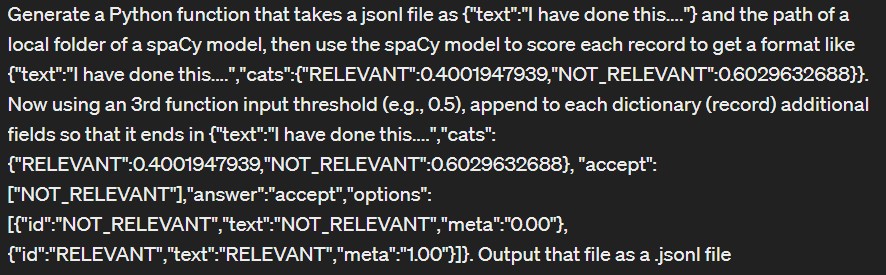
Appendix:

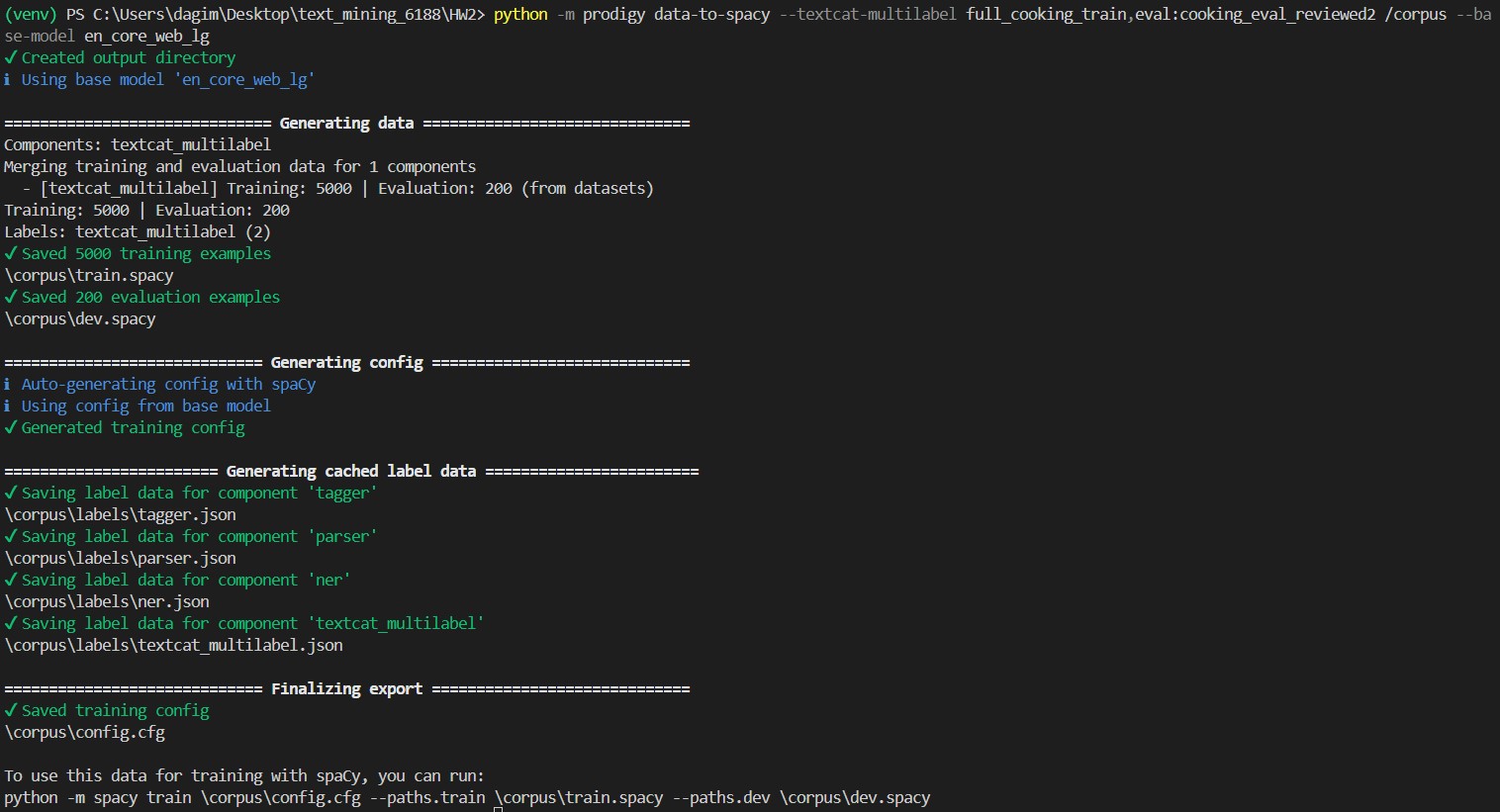
Inter-Annotated Agreement: 

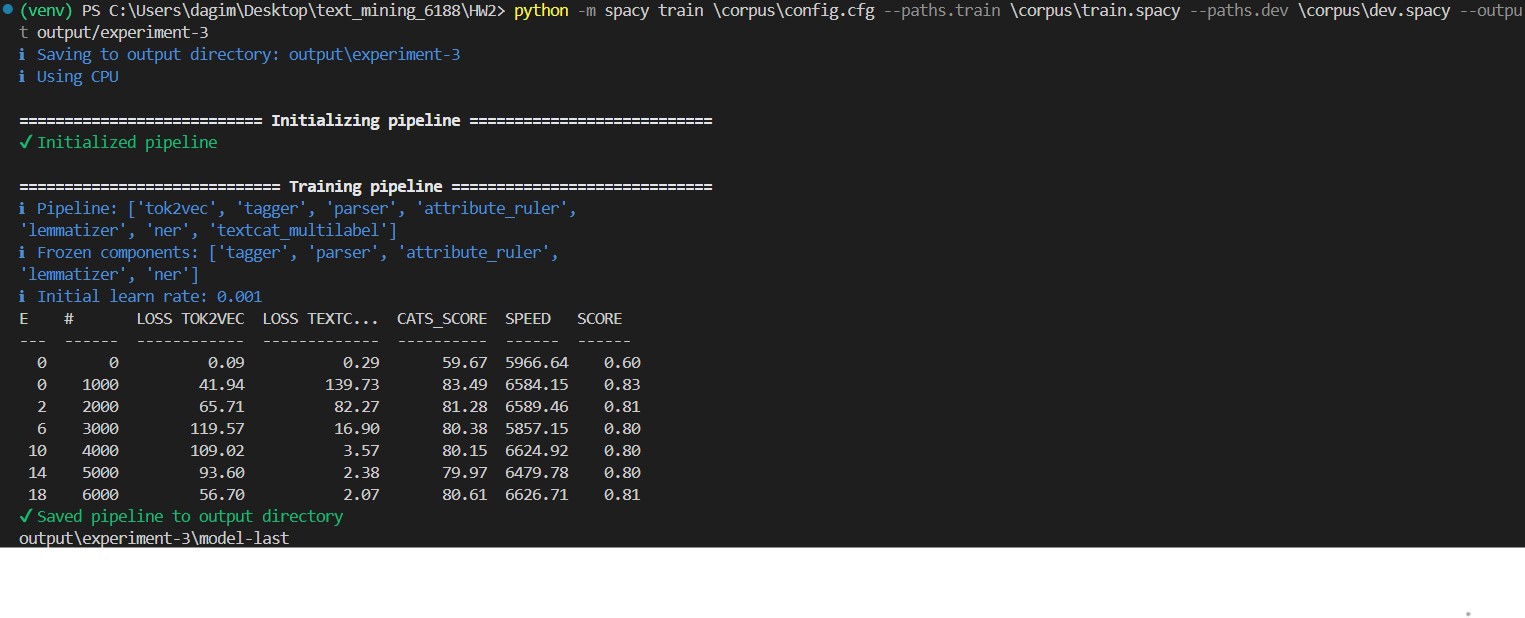
Experiment 1 Model:

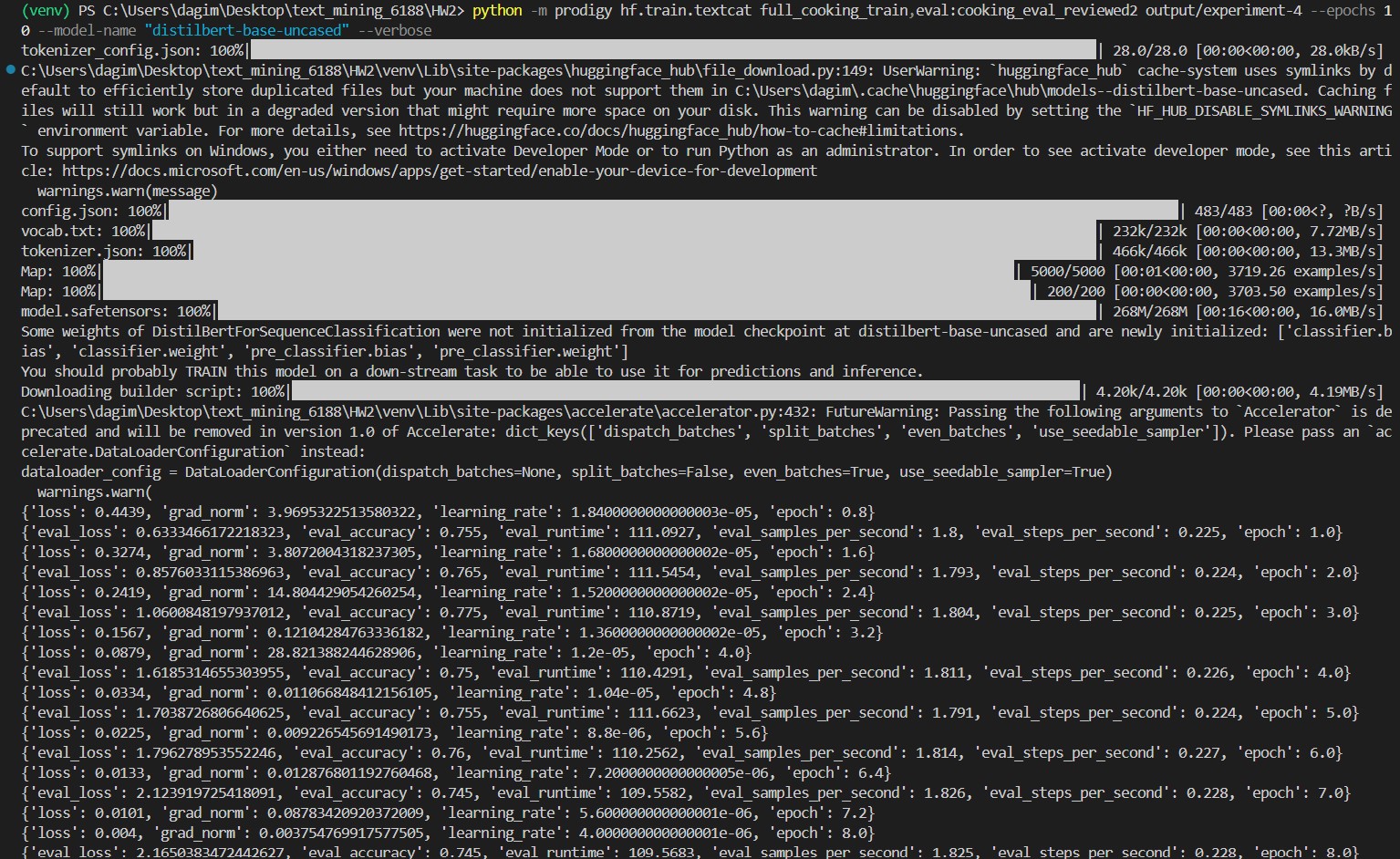
Experiment 2 Model:

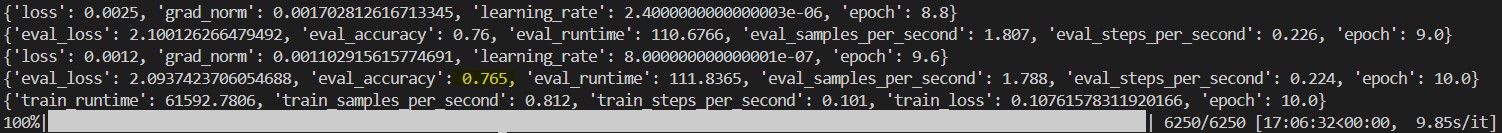
Experiment 3 ChatGPT Prompt:



Experiment 3 Model:



Experiment 4 Model:



Final Experiment Results:

|  |  |
| --- | --- |
| Experiment | Accuracy |
| 500 Manual Annotations | 0.77 |
| Full 5000 Annotations | 0.77 |
| spaCy + en\_core\_web\_lg | 0.83 |
| Hugging Face: distilbert\_base\_uncased | 0.765 |