Semantic Search - Code Challenge

Randy Reiss - 2/17/2025

Problem Statement:

“Develop a medical question-answering system utilizing the provided dataset containing medical information. The goal is to create a model that can effectively answer user queries related to medical diseases. You are free to augment the provided dataset with other medical datasets if you feel it is necessary.”

A dataset with 16406 question and answer pairs was provided

The assumption is that the objective here is recall of correct results within the top results returned

The Approach

A Bert-based embeddings model that is pre-trained on English question and answer pairs was used to convert natural language text to numeric vectors. A nearest neighbor search model was used to perform a distance calculation to compare the numeric vector for the question to the numeric vector from all the possible answers. The top 5 “closest” answers are returned to the user.

A Comparison of the performance of the semantic search using the pre-trained model and a model that is custom trained on data for this purpose. Thus, a training set and test set was randomly generated and a custom embedding model is generated by training with the training set only. Then, the test set is used to score accuracy of the correct answer occurring in the top 5 results for each question. The same test set was used to evaluate both the off-the-shelf embeddings model and the custom trained embeddings model. The results show the power of training on custom data.

The Embeddings Model:
A HuggingFace asymmetrical embedding model was selected as follows:

msmarco-MiniLM-L6-cos-v5 :
https://huggingface.co/sentence-transformers/msmarco-MiniLM-L6-cos-v5

Reason for the embeddings model selection:

1. Pre-trained on English question and answer (MS Marco) pairs for semantic search
2. The model is asymmetrical with respect to its inputs, meaning that it is specifically able to align vectors with different text input lengths. That is, it expects relatively short queries that return relatively longer answers. Thus it is able to optimize short and long text content in the vector space and this works well for semantic search as the queries are often shorter than the answers.

3. Small model size and can support inference on CPU only
4. Trainable using a custom data set and GPUs

Implementation One: Off-The-Shelf model (No Training)

- Notebook: semantic_search_wo_custom_model
- Uses the pre-trained model without additional training
- All the answers are embedded and a nearest neighbor model is complied (https://scikit-learn.org/stable/modules/neighbors.html#neighbors)
- To perform a search:
  - The embedding for the new query is calculated
  - The distance from the new query embedding is compared to the answer space and the top 5 "closest" answer vectors are selected and the associated answer text is displayed
- The test set from Implementation Two is used to measure the accuracy of the results
- Model Performance:
  - 74% recall in top 5 results
  - "test set accuracy: 0.7404021937842779 number correct: 2430"

Implementation Two: Custom Trained Embedding Model

- Training Notebook: semantic_search_train_custom_model
  - The data was randomly split into 80% training set and 20% test set
  - The "sentence_Transformer" library was used to train the model on AWS GPU EC2
  - The loss function "MultipleNegativesRankingLoss" was selected as it works best for semantic search embedding models
  - The model was trained on the training set only (and the test set saved for evaluation)
  - 10 epochs of training on a GPU compute instance (EC2) on AWS SageMaker (about an hours time to train)
  - Training was stopped after 10 epoch as the Loss value was experiencing diminishing returned (see images below)

- Inference Notebook: semantic_search_w_custom_model
  - Load the custom trained model
  - All the answers are embedded and a nearest neighbor model is complied (https://scikit-learn.org/stable/modules/neighbors.html#neighbors)
  - To perform a search:
    - The embedding for the new query is calculated
    - The distance from the new query embedding is compared to the answer space and the top 5 "closest" answer vectors are selected and the associated answer text is displayed
  - The test set is used to measure the recall/accuracy of the results
  - Model performance:
    - 97% accuracy/recall

            - test set accuracy: 0.9710542352224254   number correct: 3187

Model Evaluation and Conclusion

Off-the-shelf embedding rarely performs well for a technical subject area. The need for special training that can understand the specific meaning for technical terms and jargon make the model much more performant. This was the case for this exercise as we see the custom trained embedding model performed with a 97% recall for the correct answer being within the top 5 results returned whereas the off-the-shelf model only performed at a 74% recall.

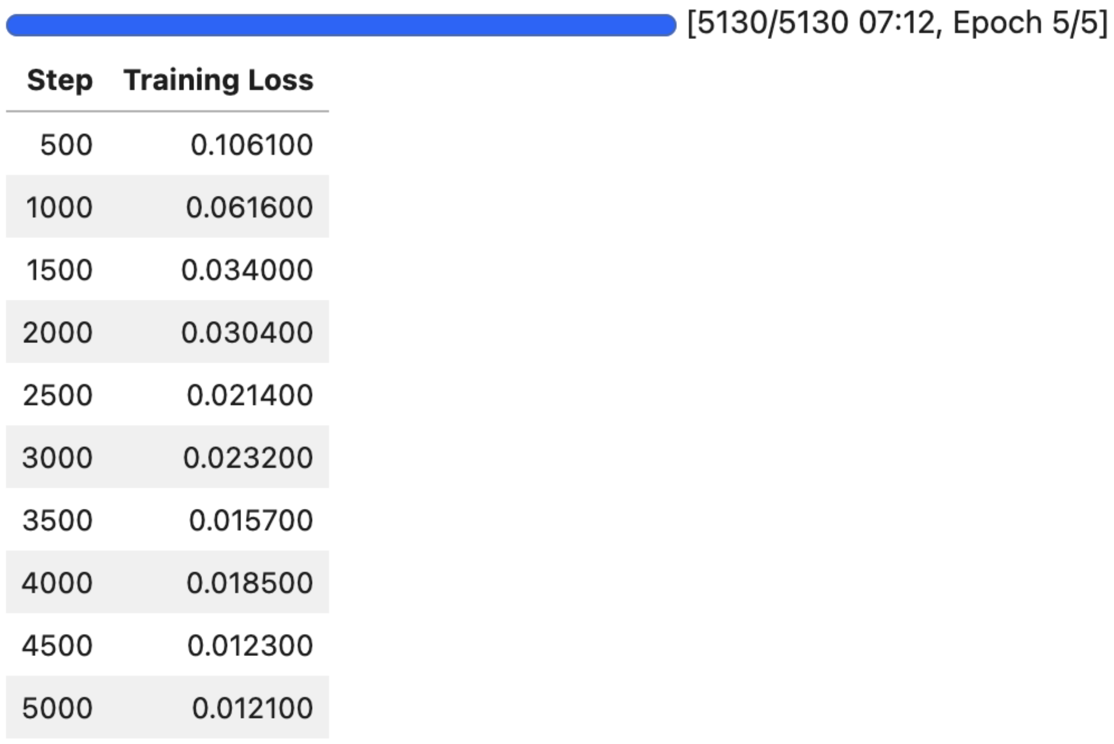
The result of the custom embeddings model is a very good recall semantic search obtaining a 97% accuracy. However, there are some improvements that could be made with more time and better compute resources.

What Could Have Been Done Better:

1. More information on the objective of the model could affect how the model is trained. It was assumed the objective was recall with the top 5 results. If precision was the objective, more training and possibly a larger embeddings model may be necessary.
2. More time can be spent on curating the training data to ensure it only contains question and answer pairs that are about the intended topic; “medical information”. It was noticed that there may be some observations that are more broad than just medical subject matter.
    a. Method that coudl be useful:
        i. LLM as a judge to weed out non-medical question answer pairs
            1. Each question and answer pair could be run through a prompt to an LLM like GPT-40 or ClaudeV3.5 Sonnet to evaluate if they are “medical information” or not
        ii. LLM to generate question based on a specific persona
            1. A LLM like GPT-4o or ClaudeV3.5 Sonnet could be used to generate additional questions to broaden the scope of the embeddings model.
            2. Different personas could be used to generate different training sets for different uses.
                a. For example:
                    i. “Questions a patient might ask a doctor”
                    ii. “Questions a doctor would ask and expert”
                    iii. “Questions a researcher would ask”
3. A larger Embedding model could be used to capture more details in training. The embedding model chosen was intended to be small enough that inference on a CPU (laptop) would be performant. If the is not a constraint, then a more complex embedding model could be used.
4. A separate hold out set that is human curated could better evaluate the performance of the model. That is, work with the stakeholders and have them put together a test set that

they would be willing to sign-off on if we reach a specific level. This solidifies an approval of the quality.

Appendix A: Model Training Loss



[5130/5130 07:12, Epoch 5/5]

Step	Training Loss
500	0.106100
1000	0.061600
1500	0.034000
2000	0.030400
2500	0.021400
3000	0.023200
3500	0.015700
4000	0.018500
4500	0.012300
5000	0.012100

[5130/5130 07:09, Epoch 5/5]

Step	Training Loss
500	0.009000
1000	0.011100
1500	0.008000
2000	0.007500
2500	0.005700
3000	0.008000
3500	0.005900
4000	0.007300
4500	0.006900
5000	0.008100