Mini Project 1-Part 2 Report

Ranking Documents

Comparison of embedding methods

1. Compare GloVe embeddings vs. Sentence Transformer embeddings
   GloVe is a quick and simple word embedding. It is great for tasks like finding similar words or solving word puzzles. On the other hand, Sentence Transformers are more advanced embeddings that understand the full meaning of a sentence. They take more computing power but do a better job at handling complex tasks.

2. Which method ranked documents better?
   The Sentence Transformer model ranks documents better. It achieved a mAP score of 0.459, while GloVe only scored roughly 0.051.

3. Did the top-ranked documents make sense?
   Yes. The top-ranked documents from Sentence Transformers made sense because they were contextually relevant to the queries, thanks to their ability to understand sentence-level meaning. In contrast, GloVe's rankings sometimes didn't make sense because it uses a simpler method, which can miss important context and lead to less relevant results.

4. How does cosine similarity behave with different embeddings?
   Cosine similarity works well with GloVe for comparing individual words, making it useful for tasks like finding synonyms or word analogies. However, GloVe struggles with sentences or context-specific meanings. On the other hand, Sentence Transformers excel at comparing entire sentences, capturing context and nuance, which makes them more accurate for complex tasks like paraphrase detection.

Observations on cosine similarity and rankings

1. Did the ranking appear meaningful?
   Yes, the rankings from Sentence Transformers made sense overall. The model understood the meaning and word choices in the query and documents. It correctly grouped documents with similar medical concepts, even if different terms were used, and took both the meaning and word variations into account for more accurate results.

2. Were there cases where documents that should be highly ranked were not?
   Yes, there were. Some relevant documents ranked lower because they used

different terms for the same concept, which made it hard for the model to match them. In addition, longer documents often diluted the key information, making it harder for the model to pick out what was most important.

3. What are possible explanations for incorrect rankings?
   Incorrect rankings are mainly due to a mismatch between the words in the query and documents. Sometimes, the model focused too much on common words without considering their context. This was worse with acronyms or specialized medical terms, as they could be misunderstood. Longer documents also caused issues, as key information got lost in the length, leading to wrong rankings.

Possible improvements

1. What can be done to improve document ranking?
   In my opinion, we can use medical-specific text processing to help the model understand medical terms better. Adding biomedical named entity recognition can help identify important medical concepts more accurately. We can also normalize document length to prevent longer documents from dominating the rankings.

2. Would a different distance metric help?
   I think that other metrics like Euclidean or Manhattan could be tested, but cosine similarity is still the best choice. It naturally adjusts for document length, so longer documents don't get unfairly ranked higher. It also works well with high-dimensional data, which fits the embeddings used here. Plus, cosine similarity is less affected by word frequency. It performs well at handling different word counts in documents.

3. Would pre-processing the queries or documents improve ranking?
   Yes, pre-processing the queries and documents could really help improve rankings. Expanding queries with medical terms would give a broader match range. Removing irrelevant stop words would clean up the queries and highlight important terms. In addition, adding weight to the most important parts of a query would make the rankings more relevant.

Fine-Tuning

Comparison of different training strategies

1. Which approach seemed to improve ranking?
   The [anchor, positive, negative] approach improved ranking more effectively. By including a negative example, the model learns not only what is similar but also what is different, helping it make better distinctions between relevant and irrelevant documents.

2. How did the model behave differently?
   With [anchor, positive] pairs, the model only learned similarities, making it less accurate for tricky cases. The other approach helped by showing both similar and different examples, creating clearer rules and improving ranking.

Impact on mAP score

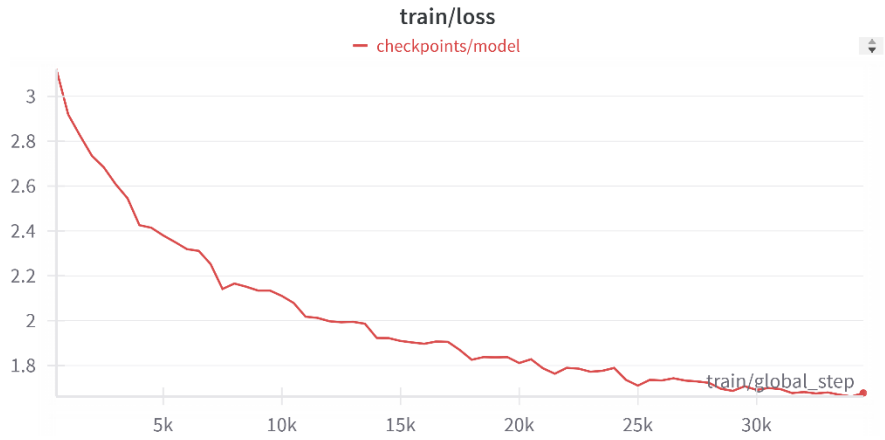
1. Did fine-tuning improve or hurt mAP score?
   Fine-tuning had a positive impact on MAP. Initially, the MAP score was 0.459. After fine-tuning, it increased to 0.464.

2. If mAP decreased, why might that be?
   It might possibly because the model overfit the data, the training data wasn't good enough, or the model didn't need fine-tuning in the first place.

3. Is fine-tuning always necessary for retrieval models?
   In my opinion, it is not always necessary. If the model is already good, fine-tuning might not help much. However, for special fields like medicine or law, or for unique data, it can make a big difference.

Observations on training loss and learning rate

1. Did the loss converge?
   Yes, it converged. Below is the screenshot of the training loss curve.



2. Was the learning rate too high or too low?
   We set the learning rate to 0.001. We think that it was a bit low. Even though the loss kept decreasing, the model took longer than expected to converge.

3. How did freezing/unfreezing layers impact training?
   Freezing all layers except the final one slightly improved the model's performance, increasing the mAP score from 0.461 to 0.464.

Future improvements

1. Would training with more negatives help?
   Yes, adding more negative examples could improve the model's ability to distinguish similar but different documents.

2. Would changing the loss function improve performance?
   Possibly. Softmax Loss could help by better separating positive and negative pairs. Other options like InfoNCE loss might also work well by improving contrastive learning.

3. Could increase the number of epochs lead to a better model?
   Yes, but only if validated properly to avoid overfitting. More epochs could help the model learn better representations, especially for complex patterns.

Contributions

Kuang-Ming Chen: rank documents, prepare training examples, fine-tune the model
Mike Huang: glove encoding, mAP function, write the report