COL764: Assignment 4

Archisman Biswas, 2021MT10254

25 November 2024

Contents

Note: All measures are given w.r.t programs run on Apple Silicon M2 chip

1 Preprocessing: Generation of Passages

The preprocessing step is designed to ensure that each training query is paired with **three** example passages. Two distinct methods for passage selection are used to form these query-passage pairs:

- **Random Passage Selection**
  Passages are randomly selected from documents marked as relevant for the given query. Each selected passage is of a length of 3 sentences to maintain uniformity. From a relevant document, only one passage is selected randomly. 3 passages are selected per query.

- **Diverse Passage Selection**
  Passages are chosen to maximize diversity with respect to their semantic content. Diversity is quantified using the cosine similarity between the passage-vector representations (Sentence Transformer embeddings) of passages. For two passages a and b, the diversity is higher if the pairwise similarity between a and b is lower than that between a and c, where c is another passage. The first passage selected is the first three sentences of the highest-ranked document in the initial retrieval set. Subsequent passages are selected to maximize dissimilarity with previously chosen passages. Here too 3 passages are selected per query.

2 Passage Selection for Prompt

The LLM used in this process is `llama3-70b-8192`, accessed via the API key provided by Groq.

`k (= 0, 1, 2, 3, 4)`-shot prompting is employed, where `k` refers to the number of `(query, passage)` pairs chosen to serve as the task context. The selected examples are presented to the LLM to provide context for answering new queries.

Prompt Format

You are a Query Answering expert. You have to output only the answer and nothing else.
Here are some example queries and their answers:
Query: `<Query1>`
Answer: `<Passage1>`

Query: `<Query2>`
Answer: `<Passage2>`
⋮
Now answer the following query. Answer should have around 3 sentences. Just return a string and nothing else:
`<Test_Query>`

If `k=0` or zero-shot, the middle part of the prompt where it is given example pairs is omitted. This prompt is designed to ensure concise, focused responses from the LLM, while leveraging the provided context of example query-passage pairs.
For passage selection earlier, there are two methods:

- **Random**
  In this, simply `k` pairs are sampled randomly from each collection from the previous step.

- **Disjoint**
  In this, the pairs are chosen randomly which are completely disjoint query with the test query. Note that stopword removal is done on both the queries before computing overlap of terms (otherwise, not getting a overlap would be almost impossible).

3 Re-Ranking of Documents

We were given an initial retrieval of 100 documents per query, which we have to now re-rank. In the previous part, a related passage was generated by the LLM, which will now be used to expand each query. Re-ranking is finally done by the BM-25 model, which scores each query-document pair and then sorts them in non-increasing order. Here the the few parts which are important for this section:

3.1 Query Expansion

The intial query is contatenated with itself 5 times, so that the length is comparable enough to the expansion string. For the expansion, a response generated by the LLM (the exact response will be clarified in the next section after experimentations) in the previous part is taken and appended to the existing query. On that, then text preprocessing is done.

3.2 Text Processing

Various text processing methods likelowercasing, removal of punctuation, removal of numerals, stopword removal, lemmatization based on which the final metrics are reported in the next section. This preprocessing is done both on queries and document to incorporate uniformity. The corpus is generated by concatening text from all documents corresponding to the top-100 documents of all queries.

3.3 Re-Ranking using BM-25

BM-25 model is used for re-ranking. The BM25 score for a document D and query Q is given by:

$$\text{BM25}(D, Q) = \sum_{t \in Q} \text{IDF}(t) \cdot \frac{f(t, D) \cdot (k_1 + 1)}{f(t, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

IDF Component

The inverse document frequency of a term t is:

$$\text{IDF}(t) = \log \frac{N - n_t + 0.5}{n_t + 0.5}$$

where:

- N: Total number of documents in the collection.
- n_t: Number of documents containing the term t.

Term Frequency Weight

The term frequency weight for a term t in document D is:

$$\frac{f(t, D) \cdot (k_1 + 1)}{f(t, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

where:

- $f(t, D)$: Term frequency of t in D.
- $|D|$: Length of the document D (number of terms).
- avgdl: Average document length in the collection.
- k_1: Term frequency saturation parameter (typically $k_1 = 1.5$).
- b: Length normalization parameter (here $b = 0.75$).

4 Results

For evaluation of the results, `nDCG@{5, 10, 50}` will be used

$$DCG[k] = \sum_{i=1}^{k} \frac{2^{G[i]}}{\log_2(i+1)}$$

$$nDCG[k] = \frac{DCG[k]}{DCG^{ideal}[k]}$$

where $G[i]$ is the relevancy of the i'th retrieved document and $DCG^{ideal}[k]$ is on the ideal vector formed on sorting the documents w.r.t relevancy in non-increasing order. The reported `nDCG@k` is the average across all test queries
For calculation, the following command will be used, with the files being of appropriate format:
`./trec_eval -m ndcg_cut.<k> <qrels> <results_file>`

4.1 Preprocessing: Passage Selection

Passage selection as stated earlier, can be done through two methods: **Random Passage Selection** and **Diverse Passage Selection**
To tune this, we first fix the prompt selection strategy to **random** and check on the varying of k (number of query-answer contexts provided to LLM)

k	nDCG[5]	nDCG[10]	nDCG[50]
0	0.5238	0.5259	0.5586
1	0.5292	0.5142	0.5376
2	0.5480	0.5206	0.5535
3	0.5288	0.5124	0.5414
4	0.5144	0.5306	0.5423
Average	0.5288	0.5207	0.5467

Table 1: Random Passage Selection

k	nDCG[5]	nDCG[10]	nDCG[50]
0	0.5120	0.5275	0.5495
1	0.5466	0.5375	0.5480
2	0.5182	0.5411	0.5444
3	0.5153	0.5189	0.5366
4	0.5291	0.5324	0.5466
Average	0.5242	0.5315	0.5440

Table 2: Diverse Passage Selection

`nDCG[5]` and `nDCG[50]` is better for the latter case, hence we choose **Random Passage Selection**

4.2 Prompt: Selection Strategy

Prompt strategy selection is done similarly by now fixing the passage selection strategy to as decided above.

k	nDCG[5]	nDCG[10]	nDCG[50]
0	0.5238	0.5259	0.5586
1	0.5292	0.5142	0.5376
2	0.5480	0.5206	0.5535
3	0.5288	0.5124	0.5414
4	0.5144	0.5306	0.5423
Average	0.5288	0.5207	0.5467

Table 3: Random Prompt Selection

k	nDCG[5]	nDCG[10]	nDCG[50]
0	0.5407	0.5340	0.5519
1	0.5519	0.5357	0.5476
2	0.5535	0.5406	0.5560
3	0.4842	0.5125	0.5381
4	0.5262	0.5258	0.5381
Average	0.5313	0.5297	0.5463

Table 4: Disjoint Prompt Selection

Since `nDCG[5]`, `nDCG[10]` is clearly better and `nDCG[50]` is very close, we choose **Disjoint Prompt Selection** strategy.
For **k**, we choose `k = 2`; as it is clearly evident from Random Passage selection and Disjoint Prompt Selection that `k=2` performs consistently better than the rest.

4.3 Text Processing: For re-ranking

In this part, some text preprocessing techniques will be explored. Initially only word-based tokenization is done using `word_tokenize` from `nltk.tokenize`

Note

All mentioned preprocessing techniques are applied to every text, i.e. queries and documents

nDCG[5]	nDCG[10]	nDCG[50]
0.5535	0.5406	0.5560

Table 5: Intial metrics

Lowercasing of characters is next tried out. Here, all characters are converted into their lowercases. This is clearly beneficial, hence **lowercasing is done**

nDCG[5]	nDCG[10]	nDCG[50]
0.6178	0.6054	0.5976

Table 6: Lowercasing characters

Removal of punctuations is next tried out. This is beneficial. Hence, it **punctuation removal is done**.

nDCG[5]	nDCG[10]	nDCG[50]
0.6309	0.6156	0.6000

Table 7: Removal of punctuation

Removal of stopwords is next tried out. The inference is not clear. Hence, it **stopword removal is not done**.

nDCG[5]	nDCG[10]	nDCG[50]
0.6254	0.6193	0.6057

Table 8: Stopword Removal

Lemmatization is next tried out. This is not beneficial. Hence, it **lemmatization is not done**.

nDCG[5]	nDCG[10]	nDCG[50]
0.6146	0.5978	0.5953

Table 9: Lemmatization

Removal of numbers is next tried out. This is beneficial. Hence, it **numbers are removed**.

nDCG[5]	nDCG[10]	nDCG[50]
0.6356	0.6182	0.6047

Table 10: Removal of numbers

At the end, the final results are presented in the next section.

5 Final Results

nDCG[5]	nDCG[10]	nDCG[50]
0.6356	0.6182	0.6047

Table 11: Final metrics

Passage Selection	LLM prompting	Re-ranking
69.4	1850.1	200.9

Table 12: Time Taken in various steps (in seconds)

5.1 Submitted files

- `[rand|div]_examples.csv`: Contains the selected passages (3 for each example query).
- `[rand|disj]_prompt.csv`: Contains the prompts per test query. For each test query, random is followed by diverse passage collections. While in each collection there are 5 rows (for 5 values of `k`). So this makes 10 rows per test query.

List of Tables