1. Please propose some updates to increase the score.

TF-IDF emphasizes important words but lacks the ability to understand the context and semantic relationships, which can limit its effectiveness in accurately capturing the semantic similarity between query embeddings and product description embeddings. To address this limitation, I initially employed Microsoft's Phi2 large language model, accessible via Hugging Face, known for its relatively lighter computational load compared to other large language models.

However, the results were suboptimal, with a MAP@10 score of 0.07, which was significantly lower than the 0.2932 achieved by the TF-IDF-based search engine algorithm. A root cause analysis can attribute this issue to two main factors:

1. The Phi2 model's final layer tensor comprises three components: the number of input sequences (batch size), sequence length, and hidden size. This structure operates at the word embedding level rather than the sentence level. Taking further averaging over the sequence length in the second component likely diluted the model's ability to effectively capture the contextual relationships within sentences.
2. Microsoft Phi2's initial input embedding uses Byte-Pair Encoding (BPE), a subword tokenization technique. Subwords generated by BPE may not always be intuitive or easily interpretable by humans, complicating the analysis of model outputs. This is particularly problematic in languages with rich inflectional systems, compounding forms, and extensive use of affixes that add significant variability to word forms. BPE might split words into subwords that do not correspond to meaningful linguistic units, leading to a loss of important morphological information.

Hence, the Microsoft's Phi2 model was not pursued any further.

**Sentence Transformers.**

To improve performance, I experimented with various sentence transformers, which are specifically optimized for sentence-level embeddings and capture semantic nuances more effectively. One of the sentence transformers for Microsoft Mini LM architecture tested yielded a MAP@10 score of 0.3495 from 0.2932, demonstrating a notable improvement over Microsoft Phi2 for this task. I believe sentence transformers are specifically fine-tuned for sentence-level similarity tasks, making them highly effective for search engines. They excel in capturing the semantic meaning of sentences, which is crucial for matching queries to product descriptions.

**Incorporating diverse and rich textual features**

Further enhancement was achieved by enriching the input text with a combination of product\_name, product\_class, category hierarchy, product\_description, and normalized\_average\_ratings, consolidated into a single string from the product information, instead of using only product\_name and product\_description. This approach leveraged more comprehensive contextual data, resulting in a further increase in MAP@10 from 0.3495 to 0.4018 as shown in Table 1. This demonstrates that incorporating diverse and rich textual features significantly enhances the model's performance in capturing semantic similarities for search engine applications

**Table 1. Comparison of MAP@10 performance metrics across different models and combined text inputs.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Sentence Transformer Model Name** | **combined\_text from product\_df table** | **Description** | **MAP@10** |
| 1 | 'sentence-transformers/all-MiniLM-L6-v2' | product\_name + product\_description | A lightweight model based on Microsoft's MiniLM architecture, designed for efficient sentence This model offers a good balance between computational efficiency and performance. It generates high-quality embeddings quickly, making it suitable for real-time search applications. | 0.3353 |
| 2 | 'sentence-transformers/all-MiniLM-L12-v2' | product\_name + product\_description | An extended version of the MiniLM model with 12 layers, offering deeper embeddings and improved performance. This model provides a balance between the lightweight nature of MiniLM models and the performance benefits of deeper architectures. It is suitable for applications needing high-quality embeddings without the full computational overhead of larger models like BERT | 0.3495 |
| 3 | 'sentence-transformers/all-mpnet-base-v2' | product\_name + product\_description | A model based on Microsoft's MPNet architecture, which combines the benefits of BERT and XLNet for improved contextual understanding. is well-suited for capturing the nuances in queries and product descriptions, making it highly effective for search engines. | 0.3379 |
| 4 | 'sentence-transformers/paraphrase-MiniLM-L6-v2' | product\_name + product\_description | A variation of the MiniLM model fine-tuned specifically for paraphrase identification tasks. This model excels at identifying paraphrases, which is beneficial for understanding queries that are rephrased in different ways. Its lightweight nature ensures efficient processing, making it practical for search engines. | 0.2856 |
| 5 | 'sentence-transformers/distilbert-base-nli-stsb-mean-tokens' | product\_name + product\_description | A distilled version of BERT, fine-tuned on Natural Language Inference (NLI) and Semantic Textual Similarity Benchmark (STSB) datasets, and uses mean pooling of token embeddings. It's a practical choice for real-time applications where resource constraints are a consideration. | 0.1711 |
| 6 (final) | 'sentence-transformers/all-MiniLM-L12-v2' | product\_name+ product\_class + category hierarchy + product\_description + normalized\_average\_ratings | The same as 2. | 0.4018 |

Of note, to leverage average rating in conjunction with cosine similarity for recommending products, I combined the normalized cosine similarity scores and normalized average ratings. This approach is based on the idea that people often consider the ratings when clicking on recommended products. A weighted sum can be used to control the influence of each metric:

Final Score = α × Normalized Cosine Similarity + (1−α) × Normalized Rating, where α is a weighting factor between 0 and 1 that adjust can be made based on how much importance we want to give to the similarity score versus the average rating. However, this method did not lead to a significant improvement in recommendation outcomes, as stated in Table 2. Hence, this approach was not pursued any further.

**Table 2. Comparison of MAP@10 performance metrics across different models and α**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Sentence Transformer Model Name** | **combined\_text from product\_df table** | α | **MAP@10** |
| 1 | 'sentence-transformers/all-MiniLM-L6-v2' | product\_name+ product\_class + category hierarchy + product\_description + normalized\_average\_ratings | 0.7 | 0.3686 |
| 2 | 'sentence-transformers/all-MiniLM-L6-v2' | product\_name+ product\_class + category hierarchy + product\_description + normalized\_average\_ratings | 0.9 | 0.3927 |
| 3 | 'sentence-transformers/all-MiniLM-L6-v2' | product\_name+ product\_class + category hierarchy + product\_description + normalized\_average\_ratings | 0.5 | 0.3330 |

1. Please implement another function that leverages the partial match count to provide a fairer assessment of performance.

Incorporating partial matches into the performance metric for Mean Average Precision at 10 (MAP@10) necessitates modifying the existing approach to account for varying degrees of relevance. Instead of using a binary relevance score (0 for Irrelevant and 1 for Exact), I defined a range of relevance scores (e.g., from 0 to 1) to reflect partial matches. For instance, from categorical relevance labels such as "exact," "partial," and "irrelevant," I mapped these categories to numerical scores (e.g., 1 for exact, 0.5 for partial, and 0 for irrelevant).

The average precision for each query is computed by summing the precision values weighted by the relevance scores and then normalizing by the sum of the relevance scores. This approach is similar to the concept used in Normalized Discounted Cumulative Gain (NDCG), even though this is not a discounted log transformation. With varying partial scores of 0.5, 0.6, and 0.7, the MAP@10 ranges from 0.5540 to 0.5904 to 0.6269, respectively, as illustrated in Table 3, which represent improvements over using a binary relevance score (0 and 1).

Of note, given the lack of a clear method for labeling ‘Partial’ in label column of label\_df table, I made a rough assumption by using 0.5, 0.6, and 0.7 as approximate values. This may represent a limitation of the approach.

**Table 3. Comparison of MAP@10 performance metrics across different models and α**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Sentence Transformer Model Name** | **combined\_text from product\_df table** | Partial Score | **MAP@10** |
| 1 | 'sentence-transformers/all-MiniLM-L6-v2' | product\_name+ product\_class + category hierarchy + product\_description + normalized\_average\_ratings | 0.5 | 0.5540 |
| 2 | 'sentence-transformers/all-MiniLM-L6-v2' | product\_name+ product\_class + category hierarchy + product\_description + normalized\_average\_ratings | 0.6 | 0.5904 |
| 3 | 'sentence-transformers/all-MiniLM-L6-v2' | product\_name+ product\_class + category hierarchy + product\_description + normalized\_average\_ratings | 0.7 | 0.6269 |

1. Please modify the code to make it more object oriented, more flexible to accommodate changes to the retrieval model, and getting it ready for production.

I refactored the code (Wayfair\_20240716.ipynb) into an object-oriented programming (OOP) format. Please refer to the attachment in the email.