Product Catalog Matching System Project

Ioanna Diamantidou

Approach

To approach a solution to the given problem, there were several steps followed:

- **Data Preview / Observations :** Observation of some samples of the given data in order to understand the typology of the request and think of the appropriate approach.
- **Data Curation :** The process of cleaning the data and bringing them to the desired format to work with, included several steps:

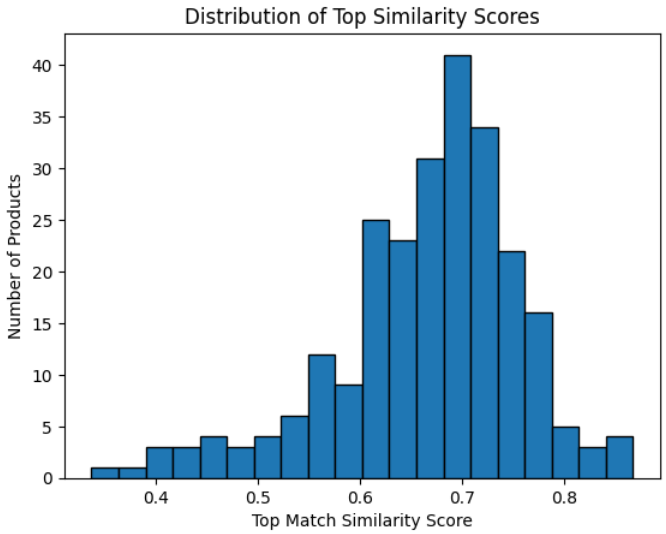
    1. Filter out filler words that do not represent any attribute but are parts of everyday speech.
    2. Normalize sizes in descriptions (e.g. extra small → xs)
    3. Create a dictionary using all words from the catalogs
    4. Compare every word from each description with the words from the dictionary and correct any typos / misspellings (this comparison is done using the Levenshtein distance which computes the letter-by-letter distance of words)
    5. Compare every word from each (typo-corrected) description with the words from the dictionary. Substitute the word with its match if the cosine similarity is over 0.75. (For words already existing in the dataset of course their best match will be themselves).
    6. Extract specific attributes (color, brand, size etc.) from the free-text descriptions.

- **Compare descriptions with catalog items:** After curating the descriptions, create embeddings of each description and each catalog item and for each description calculate the top-3 matches (based on cosine similarity score).
- **Model Evaluation :** All contextual sentence comparisons where made using 3 different models from the Sentence Transformers package :
  all-MiniLM-L6-v2, all-mpnet-base-v2 and **BAAI/bge-large-en-v1.5**
  For each model I created the distribution of the scores that correspond to each description's top match (see distributions below)

Model Evaluation

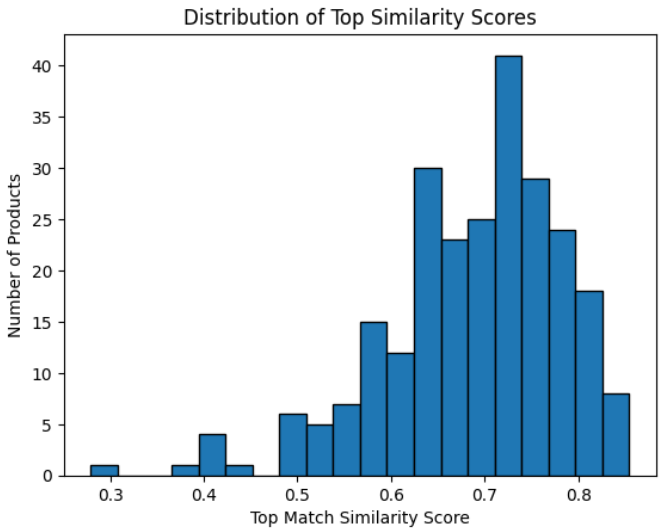
Model 1: `all-MiniLM-L6-v2`

This was the fastest of the three models, which produced somewhat satisfying results, but it was not ideal for the specific case. **Runtime : 2 minutes**



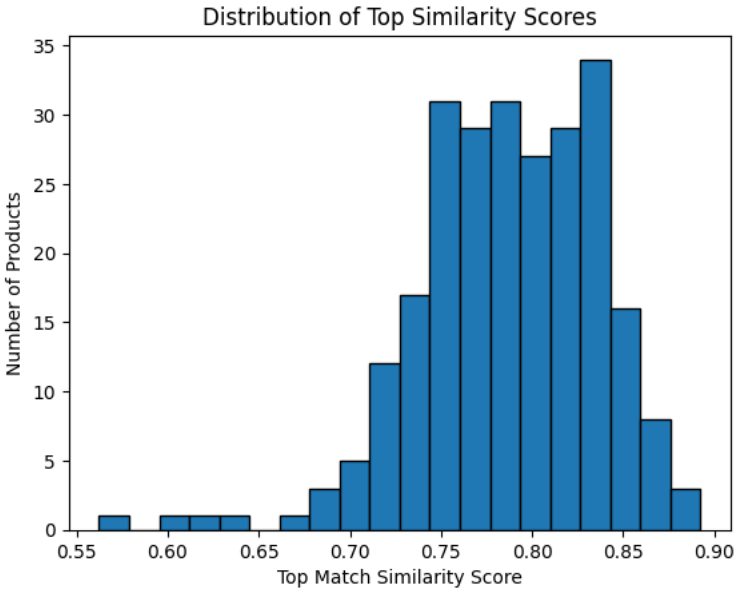
Model 2: `all-mpnet-base-v2`

The second model was also quite fast and produced slightly better results in terms of similarity scores. **Runtime : 5 minutes**



Model 3: `BAAI/bge-large-en-v1.5`

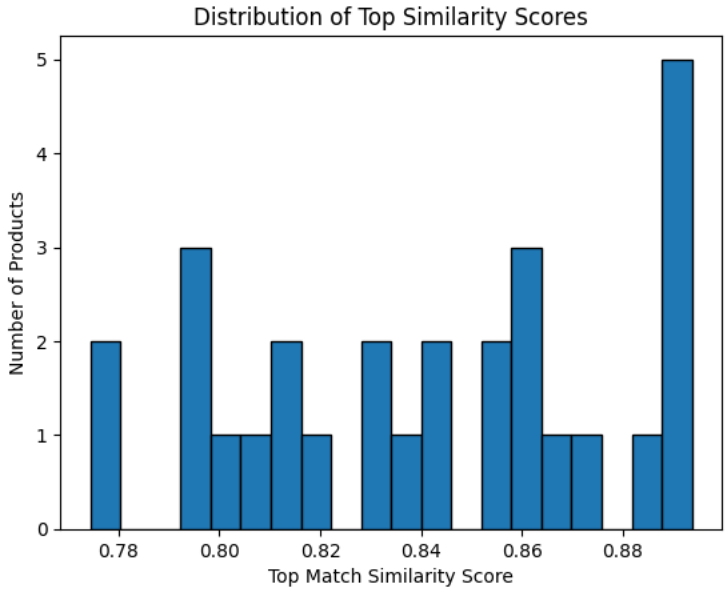
The final model was the slowest among the three but delivered the most accurate results. Given the current data scale and the fact that the model is pre-trained and does not need to be trained on each dataset separately, the additional runtime cost is minimal, so this model was chosen to achieve higher-quality matches. **Runtime : 17 minutes**



Model Testing

After selecting the optimal model, I proceeded to evaluate the algorithm using a dedicated test dataset. This dataset was constructed by examining the attributes of specific catalog items and generating corresponding free-text descriptions that incorporated some of those attributes while retaining the original SKU.
The descriptions were then matched using the previously developed algorithm, and the predicted SKUs were compared against the known ones. The results of this evaluation are presented below.

```
Test Dataset Similarity Scores Distribution:
```



Top-1 accuracy:

```
Total Descriptions: 28
Correct Matches: 26
Accuracy: 92.86%
```

Top-3 accuracy:

```
Total Descriptions: 28
Correct Top-3 Matches: 27
Recall (Top-3 Accuracy): 96.43%
```

Business Integration Plan

Integration with existing BI dashboards & KPIs recommendations

The exports generated by the product-matching algorithm (predicted SKUs, similarity scores etc.) can be directly exported to an SQL table and integrated to any existing BI dashboards. After the integration, the dashboards can display useful metrics that keep track of the overall model's performance and ensure data integrity.

KPIs Recommendations

Firstly we need to decide on a threshold that makes a matching trust-worthy (around 60%-70%), let's call it low-score threshold.

- **Low-score matching % :** The percentage of the free-text descriptions whose match got a cosine similarity score lower than the low-score threshold. This KPI helps quickly identify if the model is scoring lower than desired.

- **Average & Median cosine similarity scores**

- **Score difference between top-1 and top-2 matches**
  This KPI measures how confident the model is in its match choice. If this metric is always low it means the model is often indecisive between 2 choices.

- **User correspondence**
  Track how often the user corresponds positively to the model's suggestions

Data quality requirements for catalog maintenance

Some basic data quality requirements for the catalog would be:

- **Consistency in formatting:** meaning all catalog products should have the same format for each attribute, e.g. for sizing there should be either the 'xs-xl' format or 'extra small-extra large' format and not both.
- **No duplicate records**
- **Unique key:** There should not be 2 products with the same SKU and different attributes

Model retraining and update strategy

Since the model used is a pre-trained model, it will not need retraining. In case we notice that either the runtime is very long or the results are not satisfying we need to consider selecting some other model.

Operational Recommendations

Confidence threshold for auto-matching vs. human review

To balance accuracy with operational efficiency, a confidence threshold should be decided for automatic matching. Matches scoring above this threshold will be automatically accepted, while those below will be sent for human review. Analysis of the test dataset indicated that even matches with scores below 0.7 were often correct, suggesting that the optimal threshold would be in the range 0.7–0.8. The final value should be determined based on the available human review capacity and the desired balance between automation and accuracy.

Alert system design for low-confidence matches

Low-confidence matches can be continuously monitored through BI dashboards to track the percentage of matches falling below the confidence threshold, and alerts can be generated if this percentage increases beyond a defined limit. This way we can make sure to quickly detect and solve any data quality issues.

Scalability considerations for production deployment

Since the algorithm is based on a pre-trained model, scalability will not be a problem. The model does not need retraining for each dataset and the embeddings can be computed in batches to increase effectiveness.
For large scale deployments we could also consider using indexing to ensure fast similarity searches.