# Amazon Product Search

Github link : <https://github.com/AlexBlazee/AmazonProductSearch>

## Dataset:

I am using the Amazon’s ESCI dataset esci-data(<https://github.com/amazon-science/esci-data>) with around 1.8 million products and 2.6 million search queries

## Data preprocessing and cleaning :

### Data Sampling:

Dropped any product row which has nan value as I am planning to use pinecone as a vector DB and for the free tire version , I can upsert around 450,000 products at 1024 dimensions

English Dataset: 437953 products where products locale is ‘us’

Multilingual Dataset : Did some strategy based selection ( selected only 10 products from brands who have more than 10 products) giving 422015 products

### Cleaning:

Removed HTML script , emoticons etc.

## Embedding Models :

Build a Hybrid Search using sparse and dense embeddings

### Sparse Embedding Model:

1. BM25 from pinecone

### English dense Embedding models:

1. Voyage AI – voyage-large-2-instruct - dim (1024) model
2. AllMini - all-MiniLM-L6-v2 - dim (384) model

### Multilingual dense Embedding models:

1. Voyage AI – voyage-multilingual-2 – dim (1024) model
2. LaBSE - dim (768) model

## Vector DB:

1. Pinecone

## Re-ranker Model:

1. Jina AI’s – jina-reranker-v2-base-multilingual

## Recommendation Engine:

Recommends products in less than 10 lines of code

Capabilities:

1. Single query Search
2. Bulk Query Search in batches

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

The Bulk query searches are beneficial when using any proprietary embedding models on free-tier due to the restrictions on rate-limits. This batch -128 processing the data without hitting the rate-limits (In my case : Voyage AI embeddings for documents and queries)

## Evaluation Data:

Strategized random selection of 10K easy and 5 K hard queries for each English and Multi-lingual dataset

Strategy – select queries with products in [‘E’,’S’] – exact/substitute labels and available in the pinecone vectorized database with a threshold

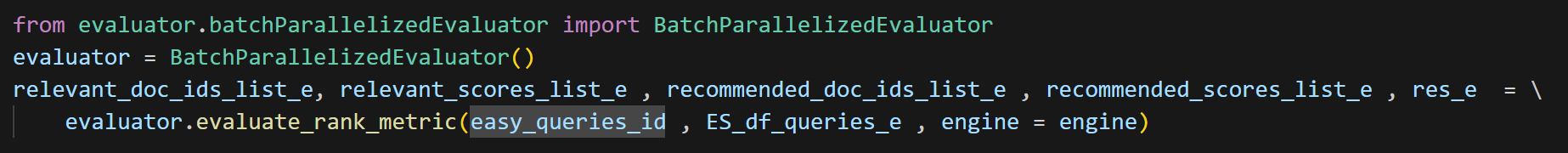
Roughly selected 30K easy and 15k hard queries and random sampled 10k easy and 5k hard

## Evaluator:

Evaluation is 1 line

Capabilities:

1. Single Query Evaluation
2. Parallelized Evaluation – query by query
3. Batch Parallelized Evaluation – query in batches

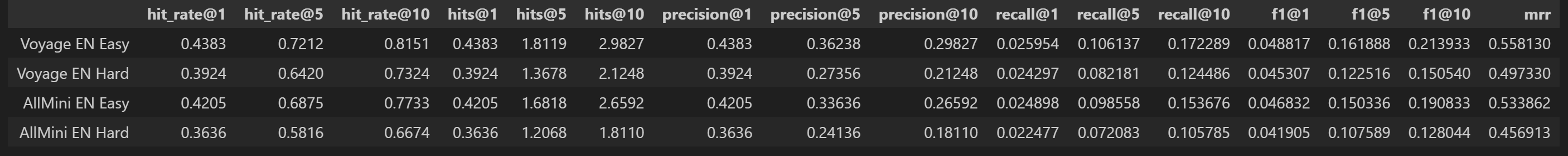


## Evaluation Metrics:

1. Hit\_rate @ (1,5,10)
2. Hits @ (1,5,10)
3. Precision @ (1,5,10)
4. Recall @ (1 , 5, 10)
5. F1 @ (1 , 5, 10)
6. MRR

## Results:

English:



Multilingual : A black screen with white text

Description automatically generated

## Stream lit and Fast API web APP:

A screenshot of a computer

Description automatically generated

A screenshot of a black and white list

Description automatically generated

## How to run :

1. Create a conda env -python 3.12 ( GPU might be required for some embedding models)
2. Install requirements.txt
3. Dataset Preprocessing
   1. Data\_products\_preprocessing.ipynb in English/Multilingual folders of Notebooks
4. Main Files :
   1. ml\_voyager\_pinecone.ipynb - multilingual voyage(Dense) bm25(Sparse) file
   2. ml\_laBse\_pinecone.ipynb - multilingual laBSE(Dense) bm25(sparse) file
   3. en\_voyager\_bm\_pinecone.ipynb – English Voyage(Dense) bm25(Sparse) file
   4. en\_allmini\_bm\_pinecone – English Allmini(Dense) bm25(Sparse) file

These files include

1. data loading
2. Pinecone setup
3. Sparse encoder setup
4. Dense encoder setup and parallel upserting (push encoded data to pinecone)
5. Hybrid Query Search Setup
6. Bulk search
7. Data\_queries\_eval\_processing.ipynb for selecting the easy/hard queries
8. Evaluations
9. Results comparison:
   1. final\_result\_comparision.ipynb

To Run the web application, use the comments at the bottom of the files api.py and app.py in given sequence