A REPORT

on

Performing Named Entity Recognition to spot named entities in an unstructured query and validating the machine learning model after optimizing it by hyperparameter tuning.

BY

Aastha Bharill - 2020B3A71794G

AT

couture.ai, Bangalore

A Practice School – I Station at

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(MAY – JULY 2022)

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)
Practice School Division

Station: couture.ai **Centre:** Bangalore

Date of Start: 30-05-2022 **Date of Submission** 22-07-2022

Title of the Project: Performing Named Entity Recognition to spot named entities in an unstructured query and validating the machine learning model after optimizing it by hyperparameter tuning

Name of Student: Aastha Bharill **ID Number:** 2020B3A71794G

Discipline: B.E. Computer Science and M.Sc. Economics

Name of Expert: Siddharth Agarwal **Designation:** ML Engineer at couture.ai

Name of PS Faculty: Dr. Ashutosh Bhatia

Key Words: Machine Learning, Natural Language Processing, Hyperparameters, Named Entity Recognition (NER), Validation, Search Engine, Autocompletion.

Project Areas: Machine Learning, Data Science, Natural Language Processing.

Signature of Student	**Signature of PS Faculty**
Aastha Bharill	**Ashutosh Bhatia**
Date: 22/07/2022	**Date: 22/07/2022**

ACKNOWLEDGEMENTS

With a keen learning spirit, I joined this project and hoped to bridge the gap between theoretical and practical learning and I have had a valuable learning experience so far. I would firstly like to thank my parents for their constant support and motivation for every endeavour I take, without which this project would not have been possible. Next, I would like to take the opportunity to thank my project mentors Mr. Siddharth Agarwal (Machine Learning Engineer, couture.ai), Mr. Tejkiran V (Machine Learning Engineer, couture.ai), and Ms. Prashanta Singh (Human Resources, couture.ai) for taking out valuable time to mentor me and provide me with the best resources to speed up my learning. A special thanks to my esteemed professor, Dr. Ashutosh Bhatia (Associate Professor at Department of Computer Science and Information Systems, Birla Institute of Technology and Science, Pilani Campus) and all the members of the Practice School Division for their generous support and encouragement. I would also like to thank my co-mentee, Shreyas Shesham for his constant support and help throughout the learning process. I would like to thank University of Michigan for providing comprehensive courses for free on coursera that greatly helped me in my learning process. Lastly, I am thankful to all my friends and teachers for believing in me. I have no words to express my gratitude to every person who has helped me along the way in making this project successful.

ABSTRACT

Our project is aimed at bettering the search engines of e-commerce platforms like Ajio. We use ML models to refine the query, extract the tokens and categorize the words in the query such that relevant results can be shown. This includes spell checking, finding related search queries, and other tasks of cleaning up and organizing the search query. This makes use of the offline corpus that includes a word corpus, numerical corpus and entity corpus. Once a query is organized into different attributes or entities, related products can be easy to find. My contribution to the project was to hyperparameter tune the existing ML models to optimize them, perform Named Entity Recognition to identify the entities present in a query, and get better results in the Auto-completion model that gives the best results to incomplete partial queries entered by users.

TABLE OF CONTENTS

INTRODUCTION

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. It gives computers the ability to become more accurate at predicting outcomes and to learn without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values. Machine Learning is commonly used in recommendation systems, fraud detection, automated stock trading, spam filtering, computer vision, malware threat detection, etc.

There are 3 primary Machine Learning Methods:

1. **Supervised Machine Learning:** In this method, data scientists supply algorithms with labeled training data and define the variables they want the algorithm to assess for correlations. Both the input and the output of the algorithm is specified.
2. **Unsupervised Machine Learning:** These algorithms discover hidden patterns or data groupings and cluster unlabeled datasets without the need for human intervention.
3. **Semi – supervised Machine Learning:** Involves a mix of the two preceding types. During training, it uses a smaller labeled data set to guide classification and feature extraction from a larger, unlabeled data set.
4. **Reinforcement Machine Learning:** A behavioral machine learning model in which the algorithm isn’t trained using sample data, it learns as it goes by using trial and error. A sequence of successful outcomes will be reinforced to develop the best recommendation or policy for a given problem.

How does ML work?

1. **Data Collection**: Gathering past data in any form suitable for processing.
2. **Data Processing/Cleaning:** Sometimes, the data collected is in the raw form and it needs to be pre-processed. Data is to be made relevant and consistent. It is to be converted into a format understandable by the machine.
3. **Division of Data:** Divide the input data into training, cross-validation and test sets.
4. **Building a Model:** Machine learning algorithms are used to make a prediction or classification. Based on some input data, which can be labelled or unlabelled, your algorithm will produce an estimate about a pattern in the data.
5. **Evaluation of the Model:** Testing our conceptualized model with data which was not fed to the model at the time of training and evaluating its performance using metrics such as F1 score, precision and recall.
6. **Model Optimization:** If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this evaluate and optimize process, updating weights autonomously until a threshold of accuracy has been met.

Hyperparameters: A hyperparameter is a parameter These are the parameters that cannot be estimated by the model from the given data. These parameters are used to estimate the model parameters and therefore their value is set before the learning process begins. A common misconception is to confuse model parameters with hyperparameters. Model Parameters are the parameters that are estimated by the model from the given data.

Hyperparameter Tuning: The process of selecting the optimal combination of hyperparameters that maximizes model performance is known as hyperparameter tuning (or hyperparameter optimization). It works by combining several trials into a single training session. Each trial is a complete run of your training application with values for your selected hyperparameters set within the limits you specify. Once you've completed this process, you'll have a set of hyperparameter values that are best suited for the model to produce optimal results. This generalisation performance is frequently estimated using cross-validation.

The three most popular Hyperparameter tuning methods are: Grid Search, Random Search, Bayesian Optimization.

In general, this process includes:

1. Define a ML model
2. Define the range of possible values for all the hyperparameters
3. Define a method for sampling hyperparameter values
4. Define an evaluative criterion to judge the model
5. Define a cross-validation method

Natural language processing (NLP) is the branch of artificial intelligence concerning the ability of computers to understand text and spoken words in the same way that humans can. NLP combines statistical, machine learning, and deep learning models with computational linguistics (rule-based modelling of human language). These technologies, when used together, allow computers to process human language in the form of text or voice data and 'understand' its full meaning, including the speaker's or writer's intent and sentiment.

NLP majorly makes use of the Natural Language Toolkit, or NLTK, an open source collection of programs, libraries, and education resources for building NLP programs. NLP is used for tasks like spam detection, machine translation, social media sentiment analysis, text summarization, etc.

NER: Named-entity recognition (NER) is a subtask of information retrieval which is the technique to extract useful and important information from raw unstructured text. It involves the task of locating and classifying named entities from unstructured text into predefined categories such as person names, organizations, locations, time expressions, monetary values, etc.

Model validation: A set of processes and activities aimed at ensuring that an ML/AI model is performing as it should, taking into account both its design goals and its end-user utility. Testing the model is an important part of validation, but validation does not end there.

Validation is an important part of model risk management because it ensures that the model doesn't cause more problems than it solves and that it follows governance guidelines. In addition to testing, the validation process examines the model's construction, the tools used to create it, and the data it used to ensure that it will run correctly.

OBJECTIVES

- **Performing Named Entity Recognition (NER) to spot named entities in an unstructured query -** This is arguably one of the most important tasks that is to be done while using NLP to analyze a query. Considering the project is based around better query identification and classification on search engines to improve the products that are shown, classifying the entities in a random query into relevant groups and searching the catalogue based on these parameters is the major part of the work. Once the entities are identified, the search query can be efficiently matched with the products that the customer might be looking for and the closest possible search results can be predicted.

- **Improving performance and Optimizing Existing ML Model using Hyperparameter Tuning –** Hyperparameter tuning has the power to completely turn a particular ML Model around and make it more optimized. In fact, Lavesson and Davidsson, in their research paper, show that tuning hyperparameters is often more important than the choice of the machine learning algorithm. The task at hand makes use of various ML models that each contain multiple hyperparameters which affect their performance. Tuning the relevant hyperparameters to get the best combination helps attain the perfect balance between computational run times and accuracy. In the existing models, despite accuracy being good, recall numbers are very low and need improvement. Another important aspect is the consideration of which hyperparameters need to be tuned.

- **Validating the Model -** Validating the results of your machine learning model is all about ensuring that you're getting the right data and that it's accurate. Validation is a crucial step in the implementation of any machine learning model because it detects problems before they become major issues. Validation techniques that are appropriate for the task help to estimate unbiased generalized model performance and provide a better understanding of how the model was trained. When your machine learning model is deployed to real-world scenarios, you want to make sure that it is correctly trained, that it outputs the correct data, and that the predictions it makes are accurate. Models that have been properly validated are capable of adapting to new scenarios in real life. Unfortunately, no single validation method will work for all machine learning models.

MAIN BODY

This project is aimed at completely restructuring the way the search engine takes and processes input queries and gives results. This task has been divided into 3 broad categories: Preprocessing, Semantic Search and Results Ranking. This makes use of the offline corpus that contains words, phrases and numerical entities.

Preprocessing involves Tokenization and Segmentation, Spell Correction and Relaxation. Tokenization is a method of dividing a piece of text into smaller units called ‘tokens’. Tokens can be words, characters, or subwords. Once this is done, it’s important to check for any possible spelling errors and to correct it so that the relevant word can be mapped and to avoid punishing the customer for the mistake. This includes probabilistic modelling as not all spelling errors are straightforward and can have multiple possibilities. For example, ‘shurts’ could either be ‘shorts’ or ‘shirts’. Therefore, multiple parameters need to be taken into consideration. These include phonetics, distance, context and frequency. An important point to note here is that these parameters are assigned specific weights and these factors are added to compute the probability of the spelling error referring to that particular word. This process is followed by relaxation, which basically ignores the words that are irrelevant and unnecessary so that the search query only consists of useful text.

Semantic Search involves Query Intent Identification, Query Rewriting and Synonyms and Result Ranking includes Attribute Scoring and User Cohort based Re-ranking. Query Intent Identification includes category identification and entity identification. There are 3 tiers of categories that we consider. L1, L2 and L3. L1 is the broad categorization of a query into Women, Men, Infants, Home & Appliances,

Stationary, etc. L2 is further categorization into the type of the product like Western Wear, Ethnic Wear, Footwear, Time wear, etc. L3 is the most specific category that further sub-categorizes L2. For example, Western Wear has products like Shirts, Tops, etc. Ethnic Wear has Kurtas, Sarees, etc. While analyzing the data, we made use of joint categories like L1L2 or L1L3 as its computationally easier. After a category is identified, the other entities in the query are identified in context to the category they belong to. For example, gold in t-shirts refers to colour whereas gold in jewelry refers to the material. This is important to match the query to the products that are relevant and have similar or same attributes. Query Intent Identification is done using Named Entity Recognition for Textual data and Comparative Rational Context for Numerical Data. Query Rewriting includes Expansion, Relaxation and Substitution. In Query Expansion, we assign synonyms and related words (these are also weighted) to the identified tokens. This helps in broadening the scope of results shown. Query Substitution is assigning possible alternatives in case the searched query is not available. For example, showing Nike products in case of unavailability of Adidas.
Results Ranking involves Attribute Scoring and User Cohort based re-ranking.

This project has 3 different ‘areas’. The offline output corpus, the online engine, and the solr engine. Preprocessing, Semantic Search and Results Ranking were a part of the online engine which use the data in the offline output corpus. Once the online engine produces the query JSON, it gets passed on to the solr engine where the products are retrieved and listed on the website/app. The offline output corpus has a word corpus, phrase corpus and numerical entities corpus. The word corpus identifies wrong words and runs probabilistic models on the possibilities of right words, synonyms, entities, substitutes, L1L2 and L1L3 categories and the token type. Numerical corpus defines the range of the numbers in the query and assigns an entity for that particular range. For example, a number like 1000 would refer to the price but 36 would refer to the size. This also differs from category to category and so is segregated on the basis of a product's L1L2 category. Entity corpus defines the attribute that the word belongs to on the basis of the category of the query.

After the query is tokenized, the tokens pass through the embedding layer first, and then passes through the LSTM Layer which uses Attention Mechanism. After more processing, we first predict L1L2. Then the earlier model along with the L1L2 model is used to predict L1L3, and these two combined together try to predict the Brand. L1L2, L1L3 and Brand are the outputs we finally generate through this model.

Now that we have established the model, it's hyperparameter tuning is vital to extract it's maximum usability. According to Lavesson and Davidsson, tuning hyperparameters is often more important than selecting a machine learning algorithm. Van Rijn and Hutter investigate the importance of specific hyperparameters as well as their interactions. Important hyperparameters, according to the authors, are parameters that explain the most variance in performance across multiple datasets. A functional ANOVA framework is used to determine the explained variance. One drawback of this approach is that the results do not directly translate into recommendations for which hyperparameters to tune. Probst et al. solved this problem by introducing hyperparameter tunability: the performance gain that can be achieved by tuning the hyperparameter. The authors compare the results of tuning the hyperparameter of interest while leaving all other hyperparameters at default values to tuning the hyperparameter of interest while leaving all other hyperparameters at default values. Their method can be used to figure out how important tuning a hyperparameter is.

To automate the hyperparameter tuning process, we first specify a set of hyperparameters and the limits to those hyperparameters' values. Then the algorithm runs those trials and fetches the best set of hyperparameters that will give optimal results. The three models that I will be considering are Grid Search, Random Search and Bayesian Optimization. Grid Search tries each set of hyper-parameters in every possible combination. We can find the best set of values in the parameter search space using this method. Because this method must try every combination in the grid size, it typically consumes more computational power and takes a long time to complete. When compared to Grid Search, the main difference in the Random Search is that instead of trying every possible combination, it selects hyperparameter sample combinations at random from grid space. As a result, it cannot guarantee that it will find the best result, such as Grid Search. However, because the computational time is so short, this search can be extremely effective in practice. In random search, the n iter value has a significant impact on the computational time and model performance as its value determines the number of times the model should look for parameters. If this value is high, there is a better chance of improving accuracy, but this comes at the cost of more computational power.

Understanding the balance between computational time and accuracy and looking at the model's structure and needs, an appropriate hyperparameter tuning method needs to be used. A problem currently being faced is very low recall numbers despite high accuracy. This should also be taken into consideration.

To gain a better understanding of the Category Prediction model built by the former team, it was important understand Convolutional Neural Networks and the various layers involved in it like Dense and LSTM. A Convolutional Neural Network typically involves two operations, which can be thought of as feature
extractors: convolution and pooling. The output of this sequence of operations is then typically connected to a fully connected layer which is in principle the same as the traditional multi-layer perceptron neural network (MLP).
After careful consideration and research, I decided to implement the Randomized Search Algorithm and the Hyperband Classifier. Owing to the complex neural model that contained multiple hyperparameters, the program kept running into errors that I could not debug with such limited experience in the field.

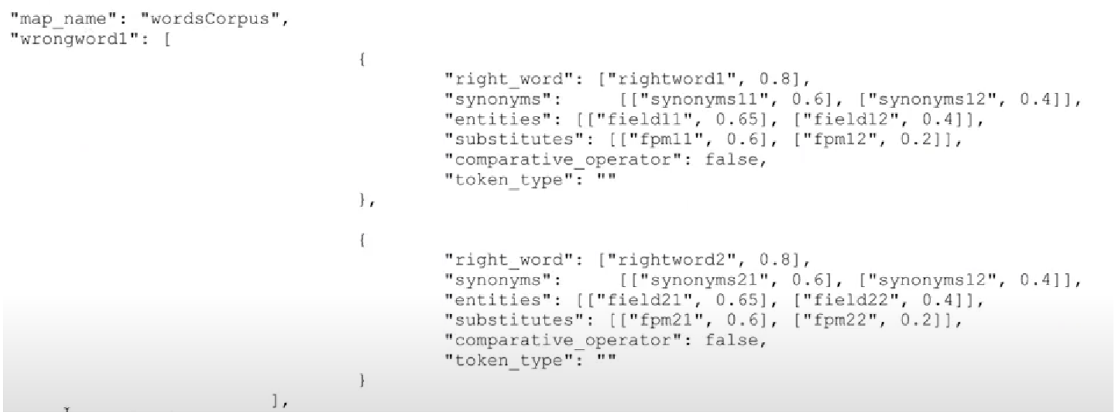
Then I got started at building an Autocomplete model that takes in Partial Queries and tests the generated BERT Output on the pre-existing Query Candidates (derived from History Data) and calculates the recall numbers. The initial model took very long to run and showed recall numbers of around 7% when tested on a subset of the two datasets. After cleaning up the data, and researching on various models to optimize the model, I made use of pre trained models from SBERT, more specifically, `sentence-transformers/all-MiniLM-L6-v2` for embedding the word vectors and used `util.semantic_search()` to compare the queries to the templates. Recall numbers increased to 10-12% using these models and computational time decreased to less than half of what it was before. Then to increase recall further, I specifically searched for generated queries that contained the prefix of the partial query and removed the others, which increased recall to 18-20%.

CONCLUSION

This project was a very good learning experience and even though I couldn’t manage to finish everything the project demanded, due to the time constraint or lack of expertise in the field, I contributed in every way I could and ultimately helped the couture.ai Semantic Search team in improving the recall on their autocomplete model and came up with useful insights while analyzing the pre-existing data sets that other members of the team had worked on.

APPENDIX

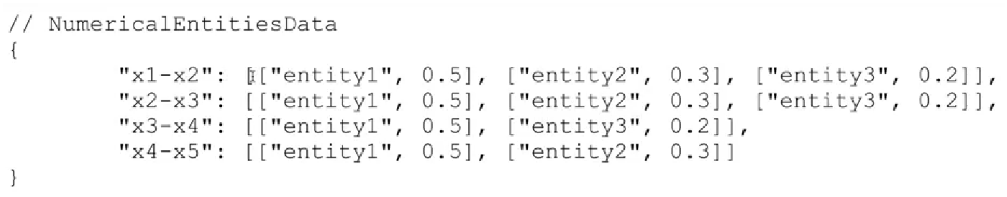
1. A sample of a Word Corpus entry:



```
"map_name": "wordsCorpus",
"wrongword1": [
                        {
                                "right_word": ["rightword1", 0.8],
                                "synonyms":     [["synonyms11", 0.6], ["synonyms12", 0.4]],
                                "entities": [["field11", 0.65], ["field12", 0.4]],
                                "substitutes": [["fpm11", 0.6], ["fpm12", 0.2]],
                                "comparative_operator": false,
                                "token_type": ""
                        },

                        {
                                "right_word": ["rightword2", 0.8],
                                "synonyms":     [["synonyms21", 0.6], ["synonyms12", 0.4]],
                                "entities": [["field21", 0.65], ["field22", 0.4]],
                                "substitutes": [["fpm21", 0.6], ["fpm22", 0.2]],
                                "comparative_operator": false,
                                "token_type": ""
                        }
                ],
```

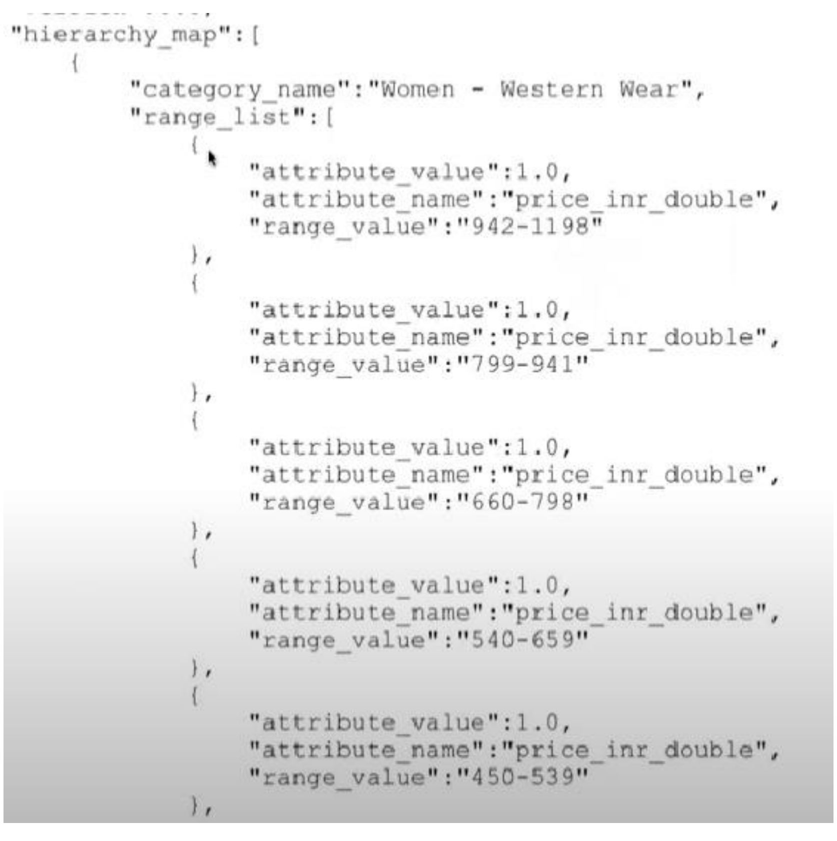
2. A sample of a Numerical Entities Corpus Entry:

Old:



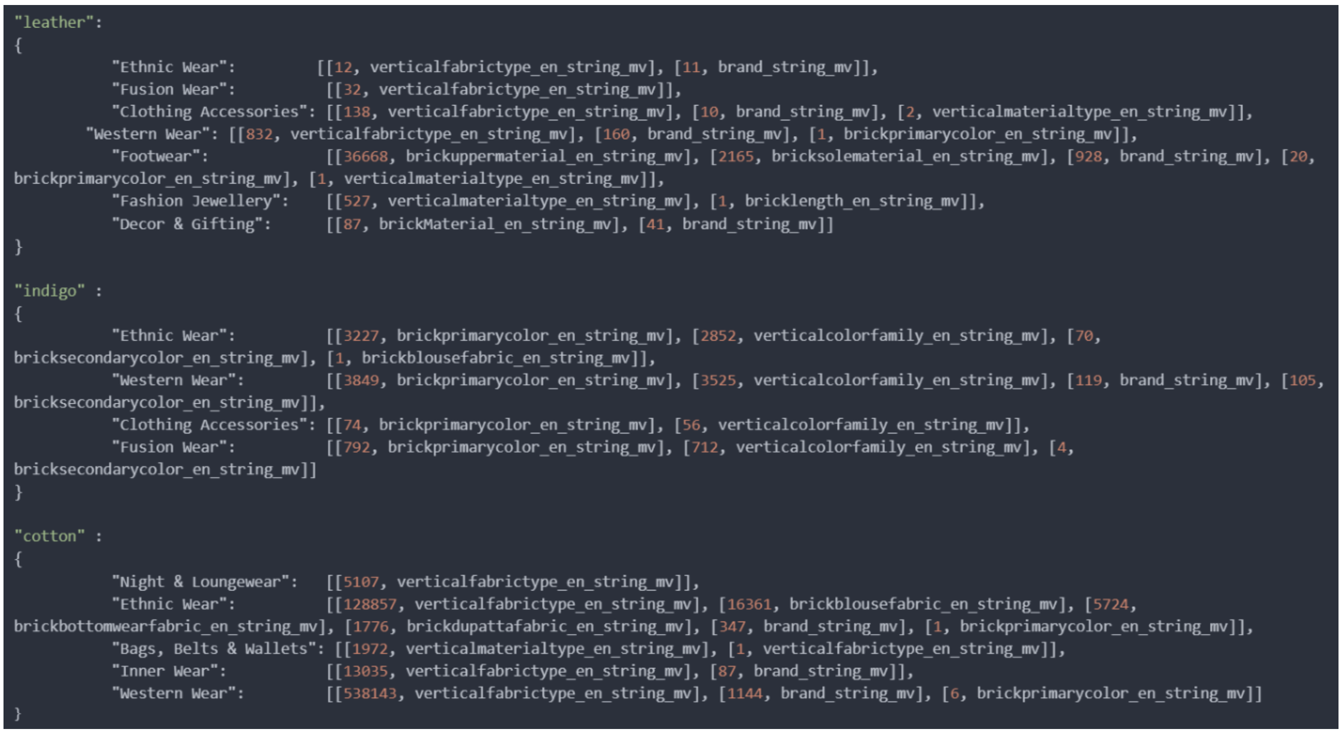
```
// NumericalEntitiesData
{
        "x1-x2": [["entity1", 0.5], ["entity2", 0.3], ["entity3", 0.2]],
        "x2-x3": [["entity1", 0.5], ["entity2", 0.3], ["entity3", 0.2]],
        "x3-x4": [["entity1", 0.5], ["entity3", 0.2]],
        "x4-x5": [["entity1", 0.5], ["entity2", 0.3]]
}
```

New:



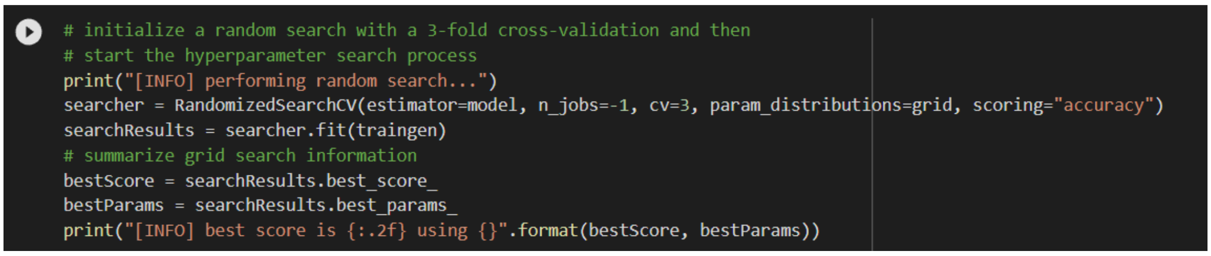
```
"hierarchy_map":[
    {
        "category_name":"Women - Western Wear",
        "range_list":[
            {
                "attribute_value":1.0,
                "attribute_name":"price_inr_double",
                "range_value":"942-1198"
            },
            {
                "attribute_value":1.0,
                "attribute_name":"price_inr_double",
                "range_value":"799-941"
            },
            {
                "attribute_value":1.0,
                "attribute_name":"price_inr_double",
                "range_value":"660-798"
            },
            {
                "attribute_value":1.0,
                "attribute_name":"price_inr_double",
                "range_value":"540-659"
            },
            {
                "attribute_value":1.0,
                "attribute_name":"price_inr_double",
                "range_value":"450-539"
            },
```

3. A sample of a Entity Corpus Entry:



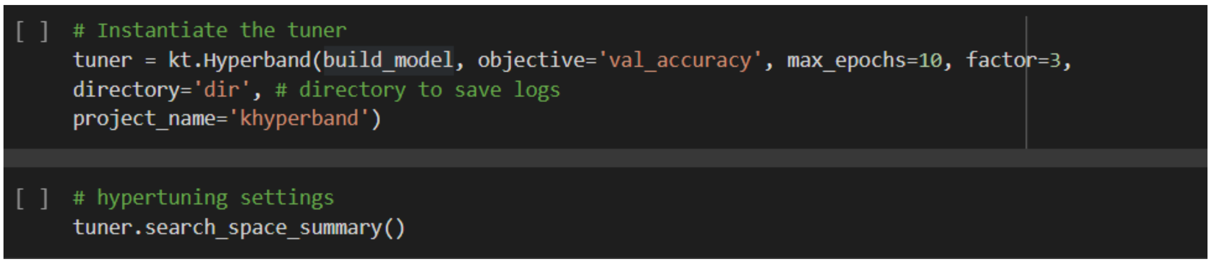
```
"leather":
{
          "Ethnic Wear":         [[12, verticalfabrictype_en_string_mv], [11, brand_string_mv]],
          "Fusion Wear":          [[32, verticalfabrictype_en_string_mv]],
          "Clothing Accessories": [[138, verticalfabrictype_en_string_mv], [10, brand_string_mv], [2, verticalmaterialtype_en_string_mv]],
       "Western Wear": [[832, verticalfabrictype_en_string_mv], [160, brand_string_mv], [1, brickprimarycolor_en_string_mv]],
          "Footwear":             [[36668, brickuppermaterial_en_string_mv], [2165, bricksolematerial_en_string_mv], [928, brand_string_mv], [20,
brickprimarycolor_en_string_mv], [1, verticalmaterialtype_en_string_mv]],
          "Fashion Jewellery":    [[527, verticalmaterialtype_en_string_mv], [1, bricklength_en_string_mv]],
          "Decor & Gifting":      [[87, brickMaterial_en_string_mv], [41, brand_string_mv]]
}

"indigo" :
{
          "Ethnic Wear":          [[3227, brickprimarycolor_en_string_mv], [2852, verticalcolorfamily_en_string_mv], [70,
bricksecondarycolor_en_string_mv], [1, brickblousefabric_en_string_mv]],
          "Western Wear":         [[3849, brickprimarycolor_en_string_mv], [3525, verticalcolorfamily_en_string_mv], [119, brand_string_mv], [105,
bricksecondarycolor_en_string_mv]],
          "Clothing Accessories": [[74, brickprimarycolor_en_string_mv], [56, verticalcolorfamily_en_string_mv]],
          "Fusion Wear":          [[792, brickprimarycolor_en_string_mv], [712, verticalcolorfamily_en_string_mv], [4,
bricksecondarycolor_en_string_mv]]
}

"cotton" :
{
          "Night & Loungewear":   [[5107, verticalfabrictype_en_string_mv]],
          "Ethnic Wear":          [[128857, verticalfabrictype_en_string_mv], [16361, brickblousefabric_en_string_mv], [5724,
brickbottomwearfabric_en_string_mv], [1776, brickdupattafabric_en_string_mv], [347, brand_string_mv], [1, brickprimarycolor_en_string_mv]],
          "Bags, Belts & Wallets": [[1972, verticalmaterialtype_en_string_mv], [1, verticalfabrictype_en_string_mv]],
          "Inner Wear":           [[13035, verticalfabrictype_en_string_mv], [87, brand_string_mv]],
          "Western Wear":         [[538143, verticalfabrictype_en_string_mv], [1144, brand_string_mv], [6, brickprimarycolor_en_string_mv]]
}
```

4. Random Search Hyperparameter Tuning



```python
# initialize a random search with a 3-fold cross-validation and then
# start the hyperparameter search process
print("[INFO] performing random search...")
searcher = RandomizedSearchCV(estimator=model, n_jobs=-1, cv=3, param_distributions=grid, scoring="accuracy")
searchResults = searcher.fit(traingen)
# summarize grid search information
bestScore = searchResults.best_score_
bestParams = searchResults.best_params_
print("[INFO] best score is {:.2f} using {}".format(bestScore, bestParams))
```

5. HyperBand Optimisation Hyperparameter Tuning

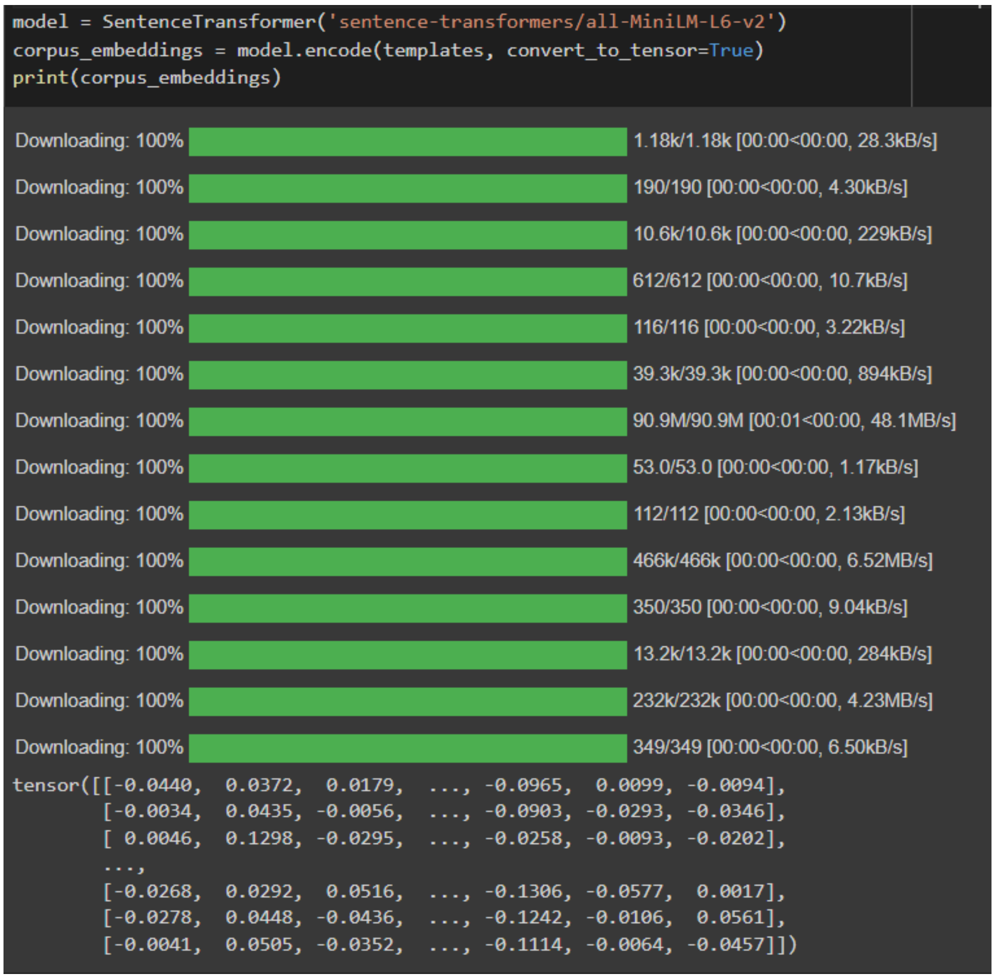


```python
# Instantiate the tuner
tuner = kt.Hyperband(build_model, objective='val_accuracy', max_epochs=10, factor=3,
directory='dir', # directory to save logs
project_name='khyperband')
```

```python
# hypertuning settings
tuner.search_space_summary()
```

Autocompletion Model:

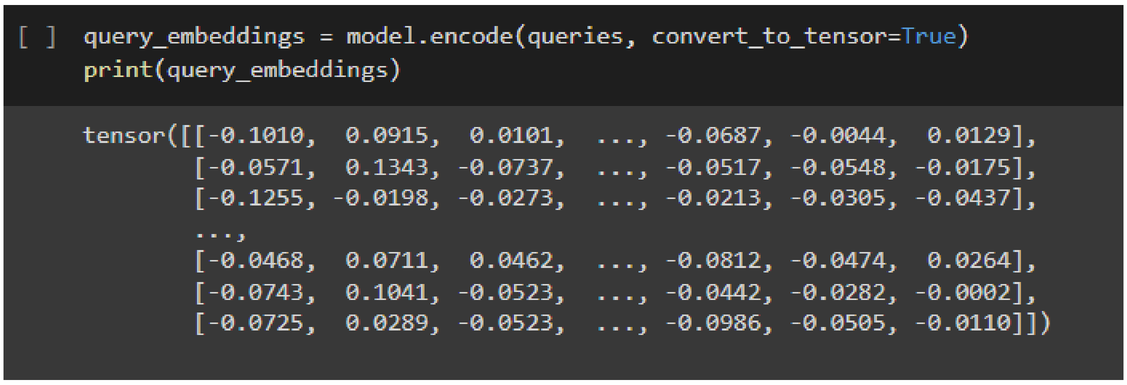
6. Using pretrained models from SBERT for embedding corpus queries



```
model = SentenceTransformer('sentence-transformers/all-MiniLM-L6-v2')
corpus_embeddings = model.encode(templates, convert_to_tensor=True)
print(corpus_embeddings)
```

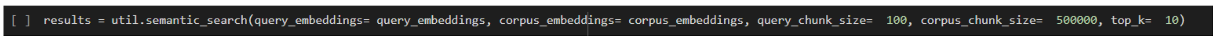
```
Downloading: 100%  1.18k/1.18k [00:00<00:00, 28.3kB/s]
Downloading: 100%  190/190 [00:00<00:00, 4.30kB/s]
Downloading: 100%  10.6k/10.6k [00:00<00:00, 229kB/s]
Downloading: 100%  612/612 [00:00<00:00, 10.7kB/s]
Downloading: 100%  116/116 [00:00<00:00, 3.22kB/s]
Downloading: 100%  39.3k/39.3k [00:00<00:00, 894kB/s]
Downloading: 100%  90.9M/90.9M [00:01<00:00, 48.1MB/s]
Downloading: 100%  53.0/53.0 [00:00<00:00, 1.17kB/s]
Downloading: 100%  112/112 [00:00<00:00, 2.13kB/s]
Downloading: 100%  466k/466k [00:00<00:00, 6.52MB/s]
Downloading: 100%  350/350 [00:00<00:00, 9.04kB/s]
Downloading: 100%  13.2k/13.2k [00:00<00:00, 284kB/s]
Downloading: 100%  232k/232k [00:00<00:00, 4.23MB/s]
Downloading: 100%  349/349 [00:00<00:00, 6.50kB/s]
tensor([[-0.0440,  0.0372,  0.0179,  ..., -0.0965,  0.0099, -0.0094],
        [-0.0034,  0.0435, -0.0056,  ..., -0.0903, -0.0293, -0.0346],
        [ 0.0046,  0.1298, -0.0295,  ..., -0.0258, -0.0093, -0.0202],
        ...,
        [-0.0268,  0.0292,  0.0516,  ..., -0.1306, -0.0577,  0.0017],
        [-0.0278,  0.0448, -0.0436,  ..., -0.1242, -0.0106,  0.0561],
        [-0.0041,  0.0505, -0.0352,  ..., -0.1114, -0.0064, -0.0457]])
```

7. Embedding partial queries



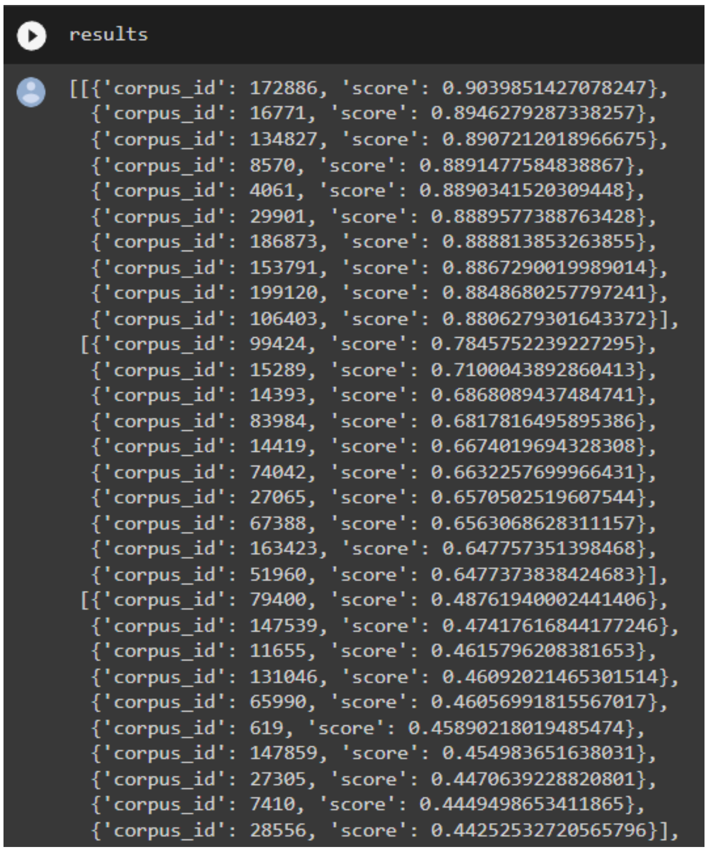
```
[ ]  query_embeddings = model.encode(queries, convert_to_tensor=True)
     print(query_embeddings)

     tensor([[-0.1010,  0.0915,  0.0101,  ..., -0.0687, -0.0044,  0.0129],
             [-0.0571,  0.1343, -0.0737,  ..., -0.0517, -0.0548, -0.0175],
             [-0.1255, -0.0198, -0.0273,  ..., -0.0213, -0.0305, -0.0437],
             ...,
             [-0.0468,  0.0711,  0.0462,  ..., -0.0812, -0.0474,  0.0264],
             [-0.0743,  0.1041, -0.0523,  ..., -0.0442, -0.0282, -0.0002],
             [-0.0725,  0.0289, -0.0523,  ..., -0.0986, -0.0505, -0.0110]])
```

8. Performing Semantic Search on corpus and query embeddings to get top_k (=10) similar queries



```
[ ]  results = util.semantic_search(query_embeddings= query_embeddings, corpus_embeddings= corpus_embeddings, query_chunk_size=  100, corpus_chunk_size=  500000, top_k=  10)
```

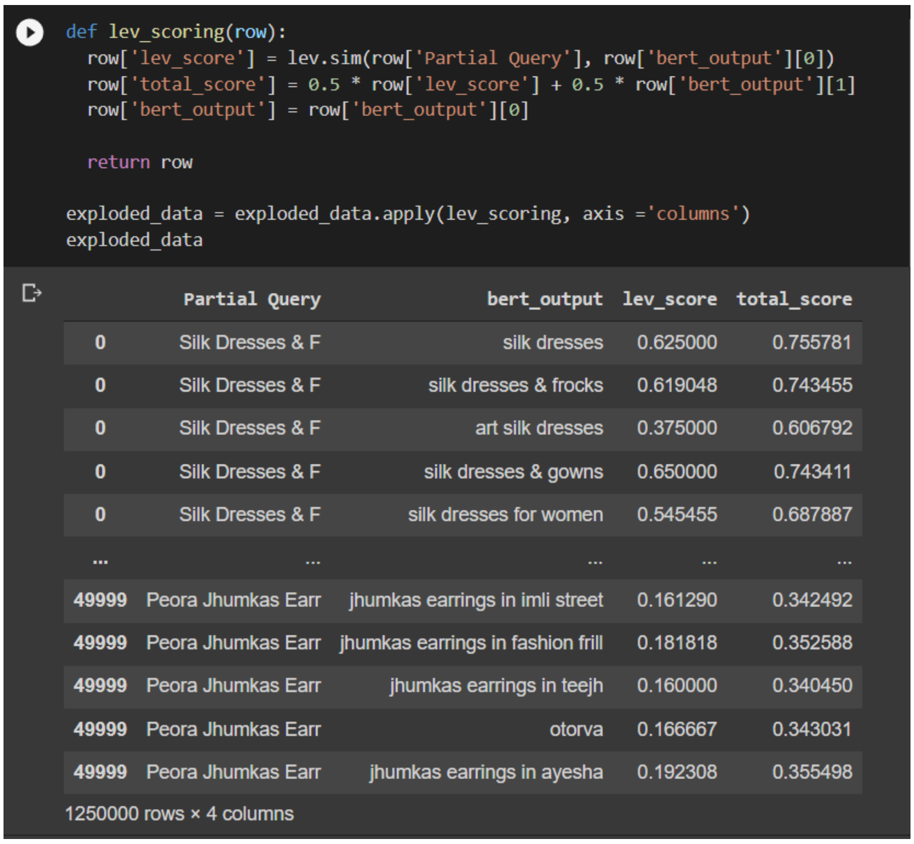
9. Snippet of generated results from above code



```
results
```

```
[[{'corpus_id': 172886, 'score': 0.9039851427078247},
  {'corpus_id': 16771, 'score': 0.8946279287338257},
  {'corpus_id': 134827, 'score': 0.8907212018966675},
  {'corpus_id': 8570, 'score': 0.8891477584838867},
  {'corpus_id': 4061, 'score': 0.8890341520309448},
  {'corpus_id': 29901, 'score': 0.8889577388763428},
  {'corpus_id': 186873, 'score': 0.888813853263855},
  {'corpus_id': 153791, 'score': 0.8867290019989014},
  {'corpus_id': 199120, 'score': 0.8848680257797241},
  {'corpus_id': 106403, 'score': 0.8806279301643372}],
 [{'corpus_id': 99424, 'score': 0.7845752239227295},
  {'corpus_id': 15289, 'score': 0.7100043892860413},
  {'corpus_id': 14393, 'score': 0.6868089437484741},
  {'corpus_id': 83984, 'score': 0.6817816495895386},
  {'corpus_id': 14419, 'score': 0.6674019694328308},
  {'corpus_id': 74042, 'score': 0.6632257699966431},
  {'corpus_id': 27065, 'score': 0.6570502519607544},
  {'corpus_id': 67388, 'score': 0.6563068628311157},
  {'corpus_id': 163423, 'score': 0.647757351398468},
  {'corpus_id': 51960, 'score': 0.6477373838424683}],
 [{'corpus_id': 79400, 'score': 0.48761940002441406},
  {'corpus_id': 147539, 'score': 0.47417616844177246},
  {'corpus_id': 11655, 'score': 0.4615796208381653},
  {'corpus_id': 131046, 'score': 0.46092021465301514},
  {'corpus_id': 65990, 'score': 0.46056991815567017},
  {'corpus_id': 619, 'score': 0.4589021801948547},
  {'corpus_id': 147859, 'score': 0.454983651638031},
  {'corpus_id': 27305, 'score': 0.4470639228820801},
  {'corpus_id': 7410, 'score': 0.4449498653411865},
  {'corpus_id': 28556, 'score': 0.44252532720565796}],
```

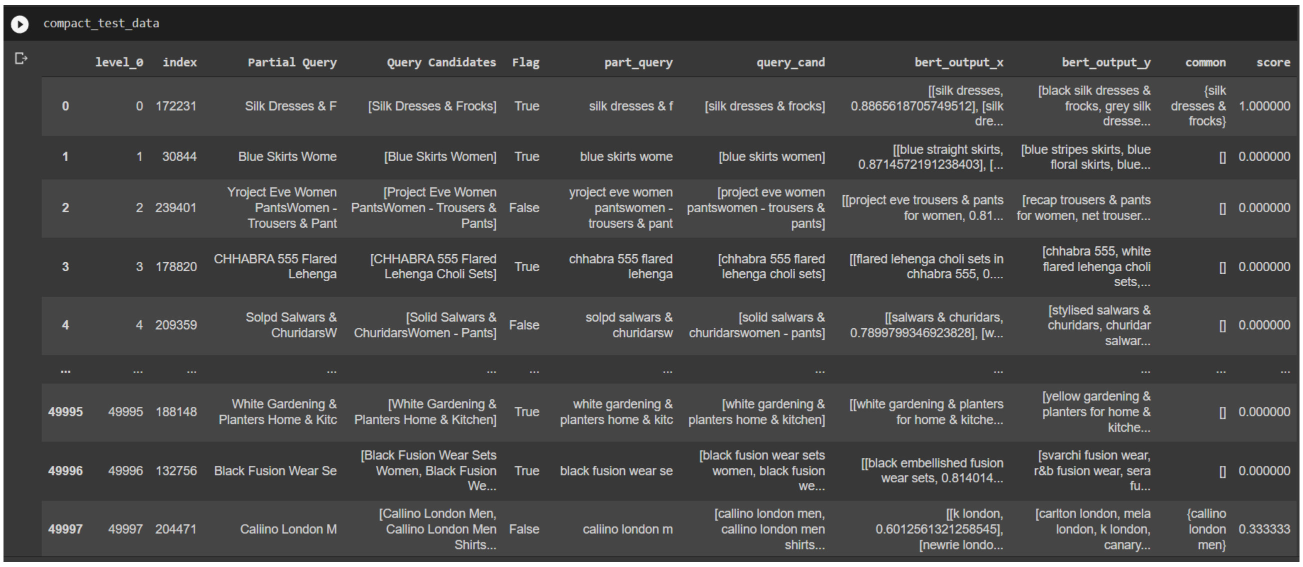
10. Calculating Total Score from Levenshtein Score for generated queries



```python
def lev_scoring(row):
  row['lev_score'] = lev.sim(row['Partial Query'], row['bert_output'][0])
  row['total_score'] = 0.5 * row['lev_score'] + 0.5 * row['bert_output'][1]
  row['bert_output'] = row['bert_output'][0]

  return row

exploded_data = exploded_data.apply(lev_scoring, axis ='columns')
exploded_data
```

```
             Partial Query                      bert_output  lev_score  total_score
    0       Silk Dresses & F                    silk dresses   0.625000     0.755781
    0       Silk Dresses & F            silk dresses & frocks   0.619048     0.743455
    0       Silk Dresses & F                art silk dresses   0.375000     0.606792
    0       Silk Dresses & F             silk dresses & gowns   0.650000     0.743411
    0       Silk Dresses & F           silk dresses for women   0.545455     0.687887
  ...                    ...                              ...        ...          ...
49999  Peora Jhumkas Earr    jhumkas earrings in imli street   0.161290     0.342492
49999  Peora Jhumkas Earr   jhumkas earrings in fashion frill   0.181818     0.352588
49999  Peora Jhumkas Earr          jhumkas earrings in teejh   0.160000     0.340450
49999  Peora Jhumkas Earr                             otorva   0.166667     0.343031
49999  Peora Jhumkas Earr         jhumkas earrings in ayesha   0.192308     0.355498

1250000 rows × 4 columns
```

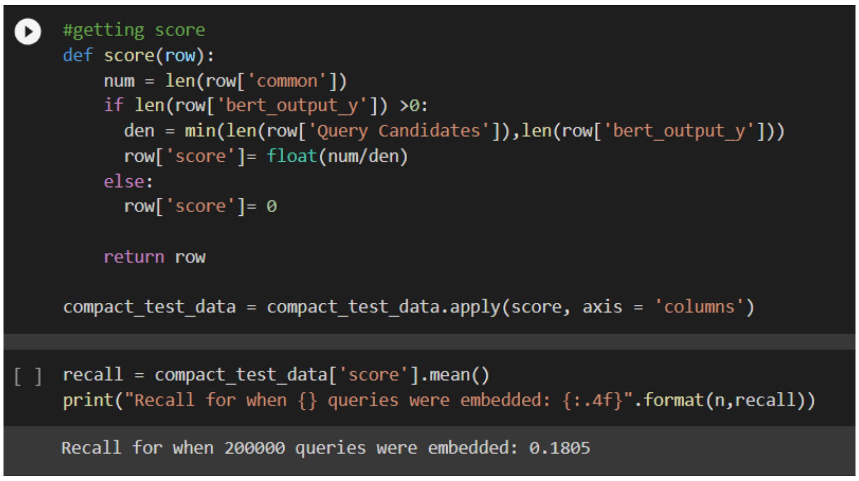
11. Final Table with common elements between Query Candidates and bert_output and the score



```
compact_test_data
```

	level_0	index	Partial Query	Query Candidates	Flag	part_query	query_cand	bert_output_x	bert_output_y	common	score
0	0	172231	Silk Dresses & F	[Silk Dresses & Frocks]	True	silk dresses & f	[silk dresses & frocks]	[[silk dresses, 0.8865618705749512], [silk dre...	[black silk dresses & frocks, grey silk dresse...	{silk dresses & frocks}	1.000000
1	1	30844	Blue Skirts Wome	[Blue Skirts Women]	True	blue skirts wome	[blue skirts women]	[[blue straight skirts, 0.8714572191238403], [...	[blue stripes skirts, blue floral skirts, blue...	[]	0.000000
2	2	239401	Yroject Eve Women PantsWomen - Trousers & Pant	[Project Eve Women PantsWomen - Trousers & Pants]	False	yroject eve women pantswomen - trousers & pant	[project eve women pantswomen - trousers & pants]	[[project eve trousers & pants for women, 0.81...	[recap trousers & pants for women, net trouser...	[]	0.000000
3	3	178820	CHHABRA 555 Flared Lehenga	[CHHABRA 555 Flared Lehenga Choli Sets]	True	chhabra 555 flared lehenga	[chhabra 555 flared lehenga choli sets]	[[flared lehenga choli sets in chhabra 555, 0....	[chhabra 555, white flared lehenga choli sets,...	[]	0.000000
4	4	209359	Solpd Salwars & ChuridarsW	[Solid Salwars & ChuridarsWomen - Pants]	False	solpd salwars & churidarsw	[solid salwars & churidarswomen - pants]	[[salwars & churidars, 0.7899799346923828], [w...	[stylised salwars & churidars, churidar salwar...	[]	0.000000
...	...	...	...	...	...	...	...	...	...	...	...
49995	49995	188148	White Gardening & Planters Home & Kitc	[White Gardening & Planters Home & Kitchen]	True	white gardening & planters home & kitc	[white gardening & planters home & kitchen]	[[white gardening & planters for home & kitche...	[yellow gardening & planters for home & kitche...	[]	0.000000
49996	49996	132756	Black Fusion Wear Se	[Black Fusion Wear Sets Women, Black Fusion We...	True	black fusion wear se	[black fusion wear sets women, black fusion we...	[[black embellished fusion wear sets, 0.814014...	[svarchi fusion wear, r&b fusion wear, sera fu...	[]	0.000000
49997	49997	204471	Caliino London M	[Callino London Men, Callino London Men Shirts...	False	caliino london m	[callino london men, callino london men shirts...	[[k london, 0.6012561321258545], [newrie londo...	[carlton london, mela london, k london, canary...	{callino london men}	0.333333

12. Scoring method used along with the final recall calculated

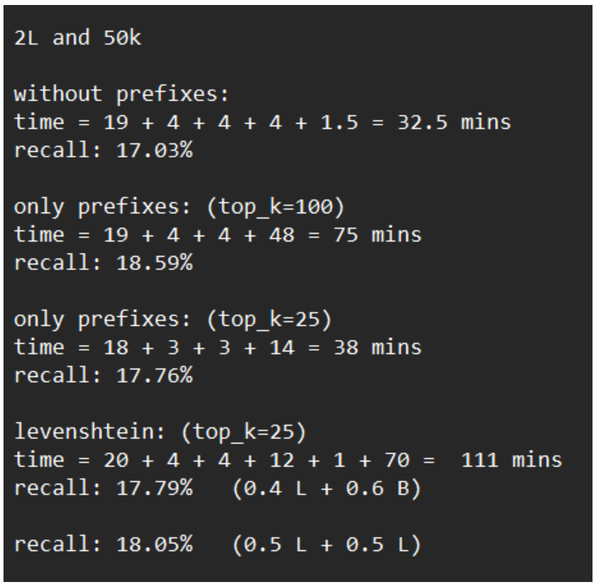


```
#getting score
def score(row):
    num = len(row['common'])
    if len(row['bert_output_y']) >0:
      den = min(len(row['Query Candidates']),len(row['bert_output_y']))
      row['score']= float(num/den)
    else:
      row['score']= 0

    return row

compact_test_data = compact_test_data.apply(score, axis = 'columns')
```

```
recall = compact_test_data['score'].mean()
print("Recall for when {} queries were embedded: {:.4f}".format(n,recall))
```

```
Recall for when 200000 queries were embedded: 0.1805
```

13. Analysing different results from various methods tried



```
2L and 50k

without prefixes:
time = 19 + 4 + 4 + 4 + 1.5 = 32.5 mins
recall: 17.03%

only prefixes: (top_k=100)
time = 19 + 4 + 4 + 48 = 75 mins
recall: 18.59%

only prefixes: (top_k=25)
time = 18 + 3 + 3 + 14 = 38 mins
recall: 17.76%

levenshtein: (top_k=25)
time = 20 + 4 + 4 + 12 + 1 + 70 =  111 mins
recall: 17.79%   (0.4 L + 0.6 B)

recall: 18.05%   (0.5 L + 0.5 L)
```

REFERENCES

Shahul ES, Aayush Bajaj (2022). Hyperparameter Tuning in Python:Complete Guide https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide

Jaswanth Badvelu (2020). Hyperparameter tuning for Machine learning models https://towardsdatascience.com/hyperparameter-tuning-for-machine-learning-models-1b80d783b946

Hilde J.P. Weerts, Andreas C. Müller, Joaquin Vanschoren (2018). Importance of Tuning Hyperparameters of Machine Learning Algorithms https://arxiv.org/pdf/2007.07588.pdf

Philipp Probst, Anne-Laure Boulesteix, Bernd Bischl (2019). Tunability: Importance of Hyperparameters of Machine Learning Algorithms https://www.jmlr.org/papers/volume20/18-444/18-444.pdf

Jeremy Jordan (2017). Hyperparameter tuning for machine learning models. https://www.jeremyjordan.me/hyperparameter-tuning/

Overview of hyperparameter tuning https://cloud.google.com/ai-platform/training/docs/hyperparameter-tuning-overview

Niwratti Kasture (2020). Why Hyper parameter tuning is important for your model ? https://medium.com/analytics-vidhya/why-hyper-parameter-tuning-is-important-for-your-model-1ff4c8f145d3

Machine Learning Model Validation – The Data-Centric Approach

https://appen.com/blog/machine-learning-model-validation/

A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

David S. Batista (2018). Convolutional Neural Networks for Text Classification

https://www.davidsbatista.net/blog/2018/03/31/SentenceClassificationConvNets/#:~:text=Convolutional%20Neural%20Networks%20(ConvNets)%20have,set%20of%20pre%2Ddefined%20categories.