TripAdvisor European Restaurant's Information

Maria Francisca Almeida
up201806398@up.pt
FEUP

Mafalda Magalhães
up201707066@up.pt
FEUP

Tomás Torres
up201800700@up.pt
FEUP

Abstract

In the current days, we come across big amounts of data and so an increasing concern to index and search efficiently appears. In this paper, an approach to creating an information retrieval system for a restaurant search system is expounded. The proposed solution includes a description of the dataset preparation, enrichment, refinement, and exploration process, as well as a detailed exposition of the information retrieval stage, from the indexation of the documents to the evaluation of the resulting search system. An overview of the used tools is also included. The final goal is to create a useful search system for restaurants that helps the users in the process of choosing a place to have a nice meal.

Keywords

Dataset, Restaurant, Review, Cuisine Style, Data Preparation, Data Analysis, Information Retrieval, Processing, Refinement, Search Engine.

1. Introduction

"Restaurants have always played an essential role in business, social, intellectual and artistic life of a thriving society" [1]. Nowadays, it's still a growing sector and clients have a bigger need to filter their never-ending choices.

The current panorama for restaurant's search systems is pretty decent in regards of the information that is able to retrieve, letting users search for names, type or location [2]. The main goal of this project is to complement this type of search systems with a search engine that allows users to search for restaurants based on ratings, reviews, cuisine styles, etc., in order to provide an easier and better experience when trying to find a restaurant that fits their preferences.

That being said, this paper aims to explain our process to extract and process information about restaurant's reviews in order to assemble an engine capable of filtering them according to the user preferences.

2. Dataset

The main dataset chosen contains the general information needed to describe restaurants, and it's reviews, from thirty one cities in Europe, gathered by TripAdvisor (TA) [3]. It was obtained by scraping TA for information about restaurants for a given city. The scraper goes through the restaurants listing pages and fulfills a raw dataset. The raw datasets for the main cities in Europe have been then curated for further analysis purposes, and aggregated to obtain this dataset.

2.1. Data Source

As for the authority of the data source, we found it in the Kaggle website while searching for a complete dataset to work with. We considered the author, Damien Beneschi, to be experienced in the area, already having other projects similar to this one. There is also a good feedback on this specific dataset in Kaggle, as well as the other works from the same author.

This dataset was a personal project of his to learn how to scrape and was published in a very well-known website, Kaggle, and he also shared the code of the program. However, since the dataset was published in 2018, the code is no longer available.

Therefore, it is concluded that it is a good data source.

2.2. Data Collection

Using the link to web page of each review in the column URL_TA, after the normalization, we tried to perform web scraping. Since this task was more expensive than we anticipated, we weren't able to perform this to every restaurant. Therefore, we performed a random scraping of 2123 restaurants reviews. This was done with the main purpose of increasing the number of reviews available so that users can better access their options when searching and choosing a restaurant. This will also help with the search engine to show which restaurants are considered 'good' or 'bad'.

2.3. Data Preparation

Initially, it was observed that the dataset had 125527 rows which was a good number to work with (table 1). However, with a closer inspection, we found some irregularities that needed to be fixed. To initiate the preparation

and cleaning process, we started by converting the ”Ranking” column to a categorical datatype and the ”Number of Reviews” column from a float to an int. Some duplicated values were also found and we kept only the first entry of each restaurant. After this, we renamed all columns by removing blank spaces and capital letters, since it would help us later to have these names normalized. Additionally we discovered that some ratings had negative values (-1), which is clearly impossible. Therefore, we replaced these values by zero. Finally, it was observed that there were several cases of missing values and, after analyzing some of these rows we decided that it did not make sense to include them in the dataset. In other words, all the lines with missing values were discarded. After this process, we ended up with approximately 75000 rows.

After this, we created a new .csv file for the reviews using only the reviews of the original dataset. In order to do this, we had to separate the values of the column ”Reviews” so that we would have one row for each different review. This row contained the restaurant id, the content of the review and its date. During this process, we found a couple of restaurants that had no reviews and we eliminated them. A new .csv file was also created for the cuisine style, using the same process.

In the end, we finished this procedure with four .csv files that contained all the information that we need to our project.

TABLE 1. NUMBER OF NON-NULL VALUES OF EACH COLUMN OF THE ORIGINAL DATASET.

Name	125 527 non-null
City	125 527 non-null
Cuisine Style	94 176 non-null
Ranking	115 876 non-null
Rating	115 897 non-null
Price Range	77 672 non-null
Number of Reviews	108 183 non-null
Reviews	115 911 non-null
URL_TA	125 527 non-null
ID_TA	125 527 non-null

2.4. Data Enrichment

The final step of the developed pipeline consisted in merging the four datasets (the original one with all the restaurant’s information, the one with the reviews obtained using the scraper, the original reviews dataset and the cuisine styles dataset) to a new .csv file. After this step, we ended up with 71595 restaurants and we converted the final .csv into a .json that was needed in the retrieval process.

2.5. Data Characterization

Throughout the analysis of the dataset, we gathered information regarding the mean value (table 2), minimum and maximum values for each of its numerical properties (ranking, rating and number of reviews).

TABLE 2. MEAN VALUE OF RATING IN EACH CITY.

City	Mean Rating
Amsterdam	4.130654
Athens	4.233831
Barcelona	4.023047
Berlin	4.150000
Bratislava	4.087699
Brussels	3.899121
Budapest	4.098214
Copenhagen	4.006950
Dublin	4.084636
Edinburgh	4.095541
Geneva	3.979210
Hamburg	4.085597
Helsinki	3.950197
Krakow	4.201651
Lisbon	4.070515
Ljubljana	4.102167
London	3.977194
Luxembourg	3.945578
Lyon	3.993521
Madrid	3.895141
Milan	3.877356
Munich	4.036678
Oporto	4.168436
Oslo	3.912037
Paris	3.981964
Prague	4.063735
Rome	4.170070
Stockholm	3.900000
Vienna	4.067200
Warsaw	4.085290
Zurich	4.036463

We also calculated the number of restaurants per city (figure 1), per inhabitant and per square kilometer (figure 2). In order to be able to do this, we had to search for some specific information, such as the population of each city and its area.

Furthermore, we also analysed the cuisine style column. We found out that there were one hundred and twenty-six different cuisine styles and we got the global (figure 3) and local (figure 4) occurrence of one of each styles.

Finally, we thought that it would be interesting to find out which cities where the best for special diets, such as vegetarianism, veganism and a gluten free diet, among others. In order to discover this, we plotted some graphics that would be useful in this analysis (figure 5).

3. Pipeline

In order to achieve a greater quality of the chosen data, various processing tasks were executed. These steps are represented in the data pipeline (figure 6).

4. Conceptual Model

The conceptual model consists of four main classes: Restaurant, Review, CuisineStyle and ReviewScraper. Each restaurant is characterized by its id, name, link, ranking, rating, price range and city. On the other hand, each review consists of a commentary and a date and each cuisine style is represented by its name. Finally, each reviews collected by the scraper is represented by the restaurant id, the date of the review, its rating, its title and the content of the review. This is represented by figure 7.

5. Collection and Indexing

For this part, the dataset was thoroughly analysed and prepared to be in the right format to use in our selected tool (Solr) and to choose the appropriate indexes that will help with the search systems defined.

5.1. Documents

As it was stated before, at the end of the Data Preparation phase, we had four datasets:

1) Restaurant's dataset
2) Cuisine Style's dataset
3) Restaurants Reviews' dataset
4) Restaurants Scraper Reviews' dataset

The first dataset contains the metadata about the restaurants themselves. The Reviews and the Cuisine Style include information about reviews and the types of food for each restaurant, respectively. Lastly, a web scrapping technique was used to gather more reviews since we only had two reviews of each restaurant and this information was kept in the Restaurants Scraper Reviews' dataset.

The information of these datasets was merged and stored in a JSON file and later imported to Solr, in order to be possible for the information to be queried upon.

5.2. Indexing Process

At the start of the indexing process, all fields were analysed to understand which ones should be indexable. Therefore, it was concluded that we didn't need to index the restaurant id and the restaurant url, since these values are unique and not relevant for the search system created. It was also determined that the number of reviews would not be indexable, since it doesn't add new information to our system (in most cases, the number of reviews doesn't match the actual number of reviews obtained).

5.2.1. Fields and Processing. The schema fields are described according to their type and whether they are indexable or not in table 3.

5.3. Schema

A schema is a configuration file that specifies what fields and field types Solr should understand when indexing new documents. These field types describe analyzers, which are pipelines that take a field's text value as input, and produce a token stream as output. This stream output is what will be indexed by the engine (i.e., what is used to match against search keywords later), but won't affect what is stored (i.e., the original value).

The schema also indicates that all attributes will be stored and that reviews and cuisine_styles are multi valued.

The indexed numerical values were defined using the default Solr field type intPointField or FloatPointField, according to the field, and the textual values were defined as a Solr TextField.

TABLE 3. SCHEMA FIELDS, RESPECTIVE TYPES AND INDEXATION.

Field	Type	Indexed
Name	restaurantType	Yes
City	restaurantCity	Yes
Cuisine Style	restaurantList	Yes
Ranking	solr.FloatPointField	Yes
Rating	solr.FloatPointField	Yes
Price Range	solr.TextField	Yes
Number of Reviews	solr.IntPointField	No
Reviews	solr.TextField	Yes
Restaurant URL	solr.TextField	No
Restaurant ID	solr.TextField	No
Good Reviews	solr.TextField	Yes
Bad Reviews	solr.TextField	Yes
Count Good Reviews	solr.FloatPointField	Yes
Count Bad Reviews	solr.FloatPointField	Yes
Count Total Reviews	solr.FloatPointField	Yes
Restaurant Sentiment Analysis	solr.FloatPointField	Yes

Finally, although Solr has a wide variety of default field types, some custom field types were created for text subjected to an analyser pipeline (table 4):

- **restaurantName -** This field applies several filters, such as the ClassicFilterFactory, which takes the output of the Classic Tokenizer and strips periods from acronyms and "'s" from possessives, the ASCIIFoldingFilter, which is responsible for the conversion of alphabetic, numeric and symbolic Unicode characters which are not in the basic Latin Unicode to block to their ASCII equivalents. This might be used to ignore accents in a word, which means that if the user misspells a certain word, the system will be able to retrieve. It also applies the LowerCaseFilterFactory, which converts any uppercase letters in a token to the equivalent lowercase token, and the KStemFilterFactory which is an alternative to the Porter Stem Filter for developers looking for a less aggressive stemmer [5].
- **restaurantCity -** This field is very similar to the restaurantName field and applies all the filters stated above.
- **restaurantList -** This field type also applies LowerCaseFilter, combined with the StandardTokenizer, that is responsible for splitting the text field into tokens, treating white space and punctuation as delimiters, and the StopFilterFactory, which discards, or stops analysis of, tokens that are on the given stop words list [5].

6. Retrieval

An Information Retrieval system deals with the organization, storage, retrieval and evaluation of information from documents. It can be used to retrieve documents that match a particular user's information needs. [8]

In order to develop a complete and efficient information retrieval system, one must primarily define a tool to be used, followed by indexation of the documents with some custom

TABLE 4. CUSTOM SCHEMA FIELDS.

Field Type	Filter and Tokenizer	Index	Query
restaurantName	ClassifiedFilterFactory	Yes	Yes
	ASCIIFoldingFilterFactory	Yes	Yes
	LowerCaseFilterFactory	Yes	Yes
	KStemFilterFactory	Yes	Yes
	StandardTokenizerFactory	Yes	Yes
restaurantCity	ClassifiedFilterFactory	Yes	Yes
	ASCIIFoldingFilterFactory	Yes	Yes
	LowerCaseFilterFactory	Yes	Yes
	KStemFilterFactory	Yes	Yes
	StandardTokenizerFactory	Yes	Yes
restaurantList	StopFilterFactory	Yes	Yes
	LowerCaseFilterFactory	Yes	Yes
	SynonymGraphFilterFactory	No	Yes
	StandardTokenizerFactory	Yes	Yes

filters for improving the search, and, lastly, evaluation of the system.

Solr has several query parsers available, such as the Standard query parser [11], which uses the operators q.op, the default operator ("OR" or "AND") and df, the default field name, the DisMax query parser [7], which is designed to process simple phrases (without complex syntax) entered by users and to search for individual terms across several fields using different weighting (boosts) based on the significance of each field, and, finally, the Extended DisMax query parser [9], which is an improved version of the Dismax query parser.

After doing some research, the conclusion was that the Extended DisMax was the more complete one and the more suitable query parser, since it has improved proximity, allows the specification of the fields that the user is allowed to query, includes advanced stop words handling, disallows the direct search on the fields and supports the specification of fields' weight.

From all the available parameters of Extended DisMax, the ones used in this Retrieval System were:

- **q -** defines the main query, which consists of the essence of the search.
- **q.op -** defines the default operators (AND and OR).
- **qf -** list of fields, each of which is assigned a boost factor to increase or decrease the chosen field importance on the query.

6.1. Tool Selection

There were some tools that were suggested for the information retrieval tasks, but Solr was the one chosen to this assignment since it meets better the project necessities and it is text-oriented.

6.2. Process

In order to evaluate the different systems' performance, four information needs were identified:

1) Which are the best restaurants of Europe?
2) Where can I find a restaurant according to my diet (Vegetarian, Gluten free, Vegan, etc.)?
3) Which one of these restaurants has the best reviews?
4) Which restaurant has the best "sentiment analysis"?

To properly evaluate the effect of filters, tokenizers and weighted fields in the query output, two different systems were made:

1) **System 1 -** a system that uses the schema described in section 5.3 and default weights.
2) **System 2 -** a system that uses the schema described in section 5.3 and weighted weights.

System 1 represents a basic search system that brings out the impact of applying Solr filters and tokenizers in the index and query time. On the other hand, the system 2 makes it possible to understand how the weighted weighs can influence the search results and can also prove how some fields are more relevant than others. Therefore, an *ad hoc* approach was followed, since the most important attributes were given a higher weight.

6.3. Boosts

The process of giving higher relevance to a set of documents over others is called boosting, Solr support at least four ways of changing the boost factors of the documents: by boosting terms, by boosting the fields where Solr search for the terms, by using the boost query parameter and by using query functions with the boost functions parameter bf [10].

6.3.1. Field and Term Boosts. On Solr interface, the qf parameter introduces a list of fields, each of which is assigned a boost factor to increase or decrease that particular field's importance in the query. Depending on the query at hand, we determined different weights for the cuisine style, rating and reviews.

6.4. Specifying Terms for the Standard Query Parser

A query to the standard query parser is broken up into terms and operators. Therefore, multiple terms can be combined together with Boolean operators to form more complex queries, as described below. We tested most of them in order to try to find out which modifications better suited our Information Retrieval System and produced better results. [11]

6.4.1. Phrase Match. The phrase match query analyzes the given text and creates a phrase query out of the analyzed text. Therefore, we can search for a set of words in some specific order [12].

6.4.2. Wildcards and Fuzziness. The fuzziness technique allows us to return documents that contain terms similar to the search term, as measured by a Levenshtein edit distance. An edit distance is the number of one-character changes

needed to turn one term into another. These changes can include changing, removing, and inserting a character and transposing two adjacent characters [13].

6.4.3. Proximity Searches. A proximity search looks for terms that are within a specific distance from one another [14].

6.5. Demo

Some of Solr features are represented in this section, such as a basic search (figure 8), a search with weighted weights (figure 9) and a search with the fuzziness technique (figure 10).

7. Search System Improvements

In this stage of the project, the goal was to improve the Search System described in the previous sections. To do so, the main topics addressed were:

1) Sentiment Analysis
2) Synonym Table
3) Solr Query Improvements

7.1. Sentiment Analysis

In order to better understand and classify the reviews and the restaurants in our dataset, two Sentiment Analysis algorithms were implemented.

7.1.1. Classify the Reviews. The first algorithm aims to classify the reviews as good reviews or bad reviews, according to the reviewer opinion. In order to do so, we used a python library, TextBlob [16], that processes textual data and evaluates the sentiment polarity of a certain text. The sentiment polarity varies between -1 an 1, meaning that all the negative values are bad reviews and all the positive values are good ones.

With these results, we were able to have two more lists of reviews - good reviews and bad reviews - apart from the list that contains the total number of reviews. We were also able to count the number of good and bad reviews of each restaurant, which is a feature that can be used to improve our future queries. The algorithm can be seen in action in figure 11.

7.1.2. Classify the Restaurants. On the other hand, the second algorithm aims to classify the restaurants as good or bad, according to the content of their reviews and their overall ranking. This analysis was executed twice, since our first algorithm did not retrieve the expected results. The first step of the first attempt was to define the stop words and to normalize the content of the reviews, by removing non alphanumeric values, put everything in lower case, split the words into a list and removing the stop words.Then a lemmatization process was applied to get adjectives, nouns, verbs and adverbs.

After that, the data obtained was split into two sets, the train set (80% of the data) and the test set (20% of the data).

Three algorithms were used to try and classify the restaurants:

- Naive-Bayes (NB)
- Linear Support Vector Machine (LSVM)
- Logistic Regression Classifier (LRC)

Although the three algorithms had somehow good results in terms of precision and accuracy, the LRC was the algorithm with the best scores. So, we decided to do some hyperparameter tuning on the LRC algorithm with the gridsearchCV library to know the best estimator that showed the highest prediction accuracy on k-fold stratified cross-validation. Then, these estimators are used to get the accuracy score on the test set.

However, this algorithm did not manage to classify the restaurants as we wished, since it ended up relying almost only on the rating of each restaurant and disregarding its reviews.

Consequently, we implemented a new algorithm based on the Sentiment Analysis of the Reviews. This algorithm classifies the Restaurants with a grade between 0 and 10. Five points of this grade correspond to the rating value and the other five are calculated by checking the percentage of good reviews of the restaurant. For example, if a Restaurant has a rating of 4.5 and has 7 good reviews in a total of 10 reviews, its grade will be: 4.5 + (7/10)*5 = 4.5 + 3.5 = 8.

This new algorithm allows us to have a more accurate measure of the quality of a Restaurant, since it computes its grade based on not only its rating, but also the reviews given by its customers. The algorithm can be seen in action in figure 9.

7.2. Synonym Table

In order to improve the search system, a list of synonyms has been created to return similar reviews based on the similarity of words used to search in a query. Some libraries [17] were used for that:

- **wordnet -** this library can list synonyms of a word based on the word sense and show all words with a similar sense;
- **stopwords -** a list of words in English that don't need to have synonyms;
- **enchant -** can check if a word exists in a specific language or not using a dictionary for that;

These libraries were used in order to separate all English words present on our dataset and and find suitable synonyms.

7.3. Solr Query Improvements

All the other improvements made it possible to enhance our queries on Solr, since we had more fields to search and apply boosters (good reviews, bad reviews, among others)

and we were able to do more specific queries using all this data. A good example of this is the use of the Restaurant Sentiment Analysis, since it enhanced significantly the retrieval of good restaurants, based on their reviews and rating.

The new field count total reviews was also a big part of our improvement, since it made possible to retrieve restaurants that had a minimum of three reviews, which meant that these restaurants had the reviews scrapped earlier, which are more complete and more trustworthy. This improvement can be seen in action in figure 8, 9 and 11.

8. Evaluation

For the evaluation, we considered the first 20 results, to rule out as relevant or non-relevant, since we consider that the first 10 results which are shown on the first page are the most important when evaluating this kind of systems. In the following results table is demonstrated this classification for the first 10 results, where 'Y' means it is relevant, 'N' means not relevant and '-' means there were no more results.

The metrics used to evaluate the results were:

- **Precision -** expresses the fraction of relevant documents from the retrieved documents;
- **Recall -** expresses the fraction of retrieved documents from the existing relevant documents.
- **AveragePrecision(AvP) -** provides a measure of quality across recall levels for a single query.
- **P@10 -** expresses the precision in the first 10 results.
- **Mean Average Precision (MAP) -** is the average of AvP and helps to better understand the quality of the system.
- **F-measure** a score that balances recall and precision for a query showing n results using a formula similar to a geometric mean.

8.1. Manual Evaluation

8.1.1. Query 1. *Which are the best restaurants of Europe?*

The user is visiting Europe and wants to find out the best restaurants in order to pick one.

- **q -** rating:5, good_reviews:good, good_reviews:excellent, restaurant_sentiment_analysis:[7 TO 10]
- **q.op -** OR
- **qf -** good_reviews$\hat{2}$ rating$\hat{0}$.5

The table 5 contains the weighted fields and the corresponded weight of this query, the table 6 contains the information need results for this query and the table 7 contains its metrics.

8.1.2. Query 2. *Where can I find a vegetarian restaurant in Amsterdam?*

The user is visiting Amsterdam and wants to find out the best vegetarian restaurant to got out for lunch.

- **q -** city:Amsterdam, cuisine_style:Vegetarian

TABLE 5. WEIGHTS OF THE FIELDS IN THE QUERY FOR THE SYSTEM 2.

Field	Weight
rating	0.5
good_reviews	2

TABLE 6. INFORMATION NEED RESULTS FOR QUERY 1.

Rank	System 1	System 2
1	0	1
2	0	1
3	0	0
4	1	1
5	1	1
6	1	1
7	1	1
8	0	1
9	1	0
10	1	1

- **q.op -** AND
- **qf -** rating$\hat{2}$

The table 8 contains the weighted fields and the corresponded weight of this query, the table 9 contains the information need results for this query and the table 10 contains its metrics.

8.1.3. Query 3. *Which one of these restaurants has the best reviews?*

The user wants to find out which one of the restaurants has the best reviews.

- **q -** good_reviews:good, good_reviews:excellent, good_reviews:best, , restaurant_sentiment_analysis:[7 TO 10]
- **q.op -** OR
- **qf -** good_reviews$\hat{2}$

The table 11 contains the weighted fields and the corresponded weight of this query, the table 12 contains the information need results for this query and the table 13 contains its metrics.

TABLE 7. METRICS FOR QUERY 1.

Metrics	System 1	System 2
AvP	0.48	0.87
P@10	0.6	0.8

TABLE 8. WEIGHTS OF THE FIELDS IN THE QUERY FOR THE SYSTEM 2.

Field	Weight
rating	2

TABLE 9. INFORMATION NEED RESULTS FOR QUERY 2.

Rank	System 1	System 2
1	0	0
2	1	1
3	1	1
4	1	1
5	1	0
6	0	0
7	1	1
8	1	1
9	0	1
10	0	1

TABLE 10. METRICS FOR QUERY 2.

Metrics	System 1	System 2
AvP	0.70	0.64
P@10	0.6	0.7

TABLE 11. WEIGHTS OF THE FIELDS IN THE QUERY FOR THE SYSTEM 2.

Field	Weight
good_reviews	2

TABLE 12. INFORMATION NEED RESULTS FOR QUERY 3.

Rank	System 1	System 2
1	1	1
2	1	1
3	1	1
4	1	1
5	1	1
6	1	0
7	0	1
8	0	0
9	1	1
10	0	1

TABLE 13. METRICS FOR QUERY 3.

Metrics	System 1	System 2
AvP	0.97	0.93
P@10	0.7	0.8

8.1.4. Query 4. *Which restaurant has the best "sentiment analysis"?*

The user wants to find out which one of the restaurants is the best, according to the sentiment analysis.

- **q -** restaurant_sentiment_analysis:[7 TO 10]
- **q.op -** OR
- **qf -** restaurant_sentiment_analysis$\hat{3}$

The table 14 contains the weighted fields and the corresponded weight of this query, the table 15 contains the information need results for this query and the table 16 contains its metrics.

TABLE 14. WEIGHTS OF THE FIELDS IN THE QUERY FOR THE SYSTEM 2.

Field	Weight
restaurant_sentiment_analysis	3

TABLE 15. INFORMATION NEED RESULTS FOR QUERY 4.

Rank	System 1	System 2
1	1	1
2	1	1
3	0	1
4	0	1
5	1	1
6	1	1
7	1	1
8	0	0
9	0	0
10	1	1

TABLE 16. METRICS FOR QUERY 4.

Metrics	System 1	System 2
AvP	0.76	0.98
P@10	0.6	0.8

8.2. Discussion

TABLE 17. MEAN AVERAGE PRECISION FOR EACH SYSTEM.

System	MAP
System 1	0.73
System 2	0.86

As expected, the mean average precision is higher in System 2 in half of the cases, which is the system that combines the schema with the field weights, and even when its smaller, is closer to the results of the System 1 then

on the other way around. Consequently, the mean average precision for each system is higher in system 2 (table 17) mostly because of the improvements that we made on our queries and on our fields, which means that the weight of some specific fields is now able to improve the result of the search significantly. Therefore, the right attributes to be weighted must be carefully chosen.

As a consequence of this results, it is possible to, in the future, keep exploring the new fields and improving the weighting system and obtain better results, since the schema options were well analysed and are used the ones that better suit the needs of this search system.

Besides, by analysing the Precision Recall graphs, the precision was higher in the system with weighted weights and it normally increases with the increase of the Recall metric (Figures 12, 13, 14, 15, 16, 17, 18 and 19).

9. Conclusion

In this paper, the developed information processing and retrieval process is meticulously explained, from the data gathering, cleaning and preparation phase to the assessment of the created retrieval system's quality.

Throughout this work, the datasets were well analysed and studied in order to conclude the appropriate data cleaning and preparation tasks to perform in order to prepare the information for the intended search tasks.

In a second stage of the process, it was developed an indexing process and several purposeful information needs were carefully conceived and used to evaluate and compare the developed retrieval systems.

The results obtained, even with some discrepancies, can prove that a well formulated schema and a set of attributes with different weights according to our needs have a big impact on the quality of the search system. It points out that the combination of schema with weighted fields brings better results, but still has room for improvement, as the mean average precision is still approximately 86%.

As a note of this stage, it is pointed that throughout this experience with Solr was concluded that it is not an easy tool to work with, since it presents some cons as it has poor documentation, is not user friendly and intuitive and it has some limitations regarding the attribution of weights to nested documents. This lead to a slightly slower learning curve than expected and a lot of back and forward during this phase.

Improving the search system was interpreted as improving the results for the tested queries so that users get better restaurants' recommendations according to their preferences and queries. According to the results obtained while applying the sentiment analysis algorithms and the synonym table, we can consider that these improvements were successful and our search system works in the way it was supposed to.

As a final note, the team is proud of the work done in this project, having retained a lot of knowledge about search systems and how they are optimized which is not recognizable when used as the end-user.

9.1. Future Work

Having in consideration the entirety of this work, it is considered relevant to undertake the following steps to enhance the developed retrieval system:

- Implementing a user friendly interface so that users can research a place to eat according to their preferences, diet, price, location, etc.
- Refine the synonym list. Our synonym table is based in the English dictionary so reviews written in other languages are not included in the search results.
- Implement a recommendation system that gives suggestions of restaurants to users according to their preferences.

10. Revisions Implemented

In the final version of this report, a few revisions were made and thus the changes that were made are explained in this section. The abstract was actualized to contain information about the whole project. In section 5, a small introductory text was added about the process of collection and indexing and in subsection 5.1 more information of the datasets used was added. Finally, the results of the evaluation were updated and amended and some corrections were made to the tables and images descriptions.

You can access the GitHub repository with all this information in https://github.com/mafmagalhaes19/pri-project.

References

[1] Feldman, Eli (2015), *Why the Restaurant Industry is the Most Important Industry in Today's America*

[2] Find your perfect restaurant. (2022) Retrieved 10 December 2022, from https://www.tripadvisor.ie/Restaurants

[3] TripAdvisor Restaurants Info for 31 Euro-Cities. (2022) Retrieved 19 September 2022, from https://www.kaggle.com/datasets/damienbeneschi/krakow-ta-restaurans-data-raw

[4] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[5] Solr Filter Descriptions. (2022) Retrieved 13 November 2022, from https://solr.apache.org/guide/6_6/filter-descriptions.html

[6] Solr Standard Query Parser. (2022) Retrieved 11 November 2022, from https://solr.apache.org/guide/6_6/the-standard-query-parser.html

[7] Solr Dismax Query Parser. (2022) Retrieved 11 November 2022, from https://solr.apache.org/guide/6_6/the-dismax-query-parser.html

[8] Information Retrieval Systems. (2022) Retrieved 10 December 2022, from https://www.sciencedirect.com/topics/computer-science/information-retrieval-systems

[9] Solr Extended DisMax Query Parser. (2022) Retrieved 11 November 2022, from https://solr.apache.org/guide/6_6/the-extended-dismax-query-parser.html

[10] Solr Boost. (2022) Retrieved 12 November 2022, from https://medium.com/@pablocastelnovo/if-they-match-i-want-them-to-be-always-first-boosting-documents-in-apache-solr-with-the-boost-362abd36476c

[11] The Standard Query Parser Solr. (2022) Retrieved 11 December 2022, from https://solr.apache.org/guide/8_7/the-standard-query-parser.html#specifying-terms-for-the-standard-query-parser

[12] Phrase Match. (2022) Retrieved 12 November 2022, from https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-match-query-phrase.html

[13] Solr Fuzziness. (2022) Retrieved 12 November 2022, from https://solr.apache.org/guide/6_6/the-standard-query-parser.html#TheStandardQueryParser-FuzzySearches

[14] Solr Proximity Search. (2022) Retrieved 12 November 2022, from https://solr.apache.org/guide/6_6/the-standard-query-parser.html#TheStandardQueryParser-ProximitySearches

[15] NLP: A Complete Sentiment Classification on Amazon Reviews. (2020) Retrieved 28th November 2022 from https://erleem.medium.com/nlp-complete-sentiment-analysis-on-amazon-reviews-374e4fea9976

[16] TextBlob: Simplified Text Processing. (2020) Retrieved 11 December 2022 from https://textblob.readthedocs.io/en/dev/

[17] How to get synonyms/antonyms from NLTK WordNet in Python? https://www.geeksforgeeks.org/get-synonymsantonyms-nltk-wordnet-python/amp/

Appendix

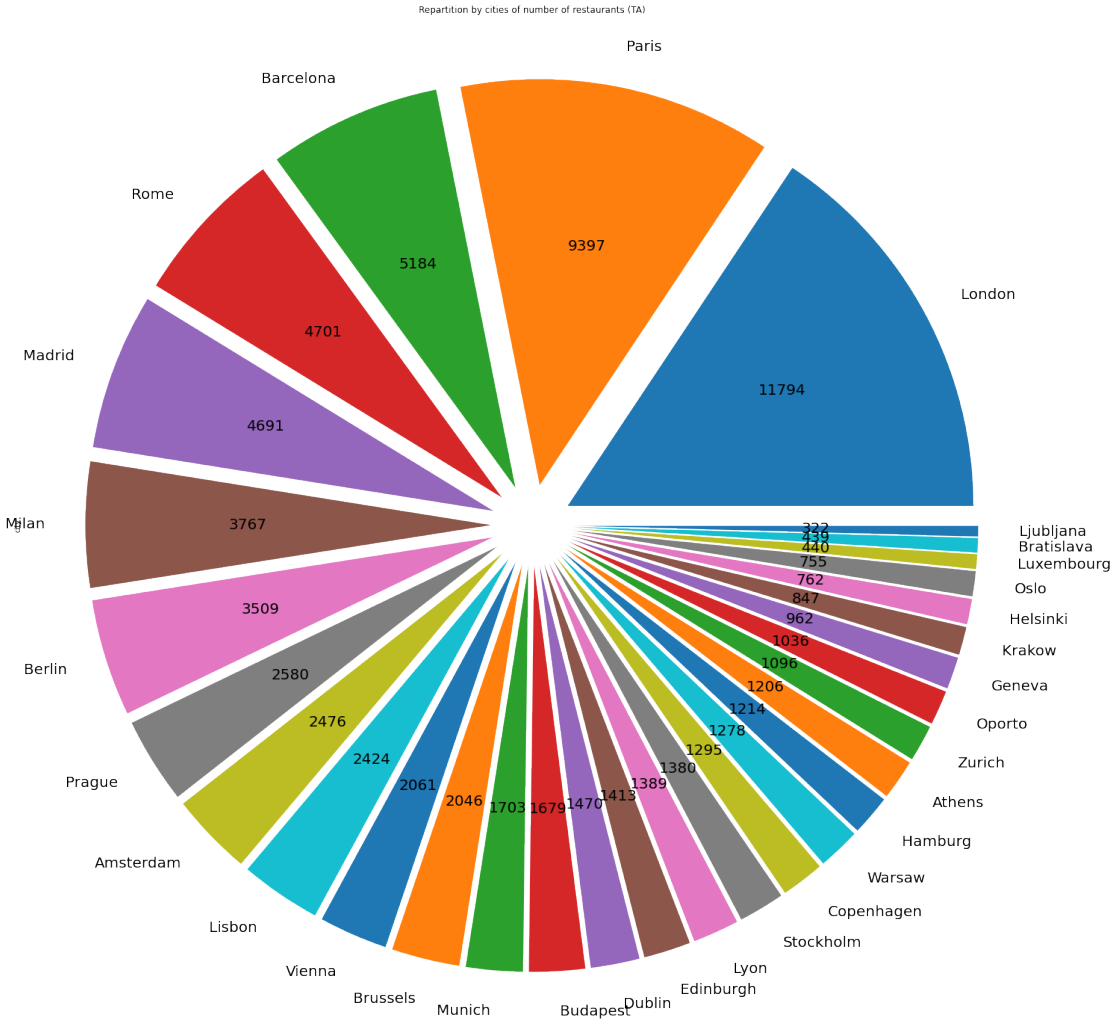

Figure 1. Number of restaurants per city.

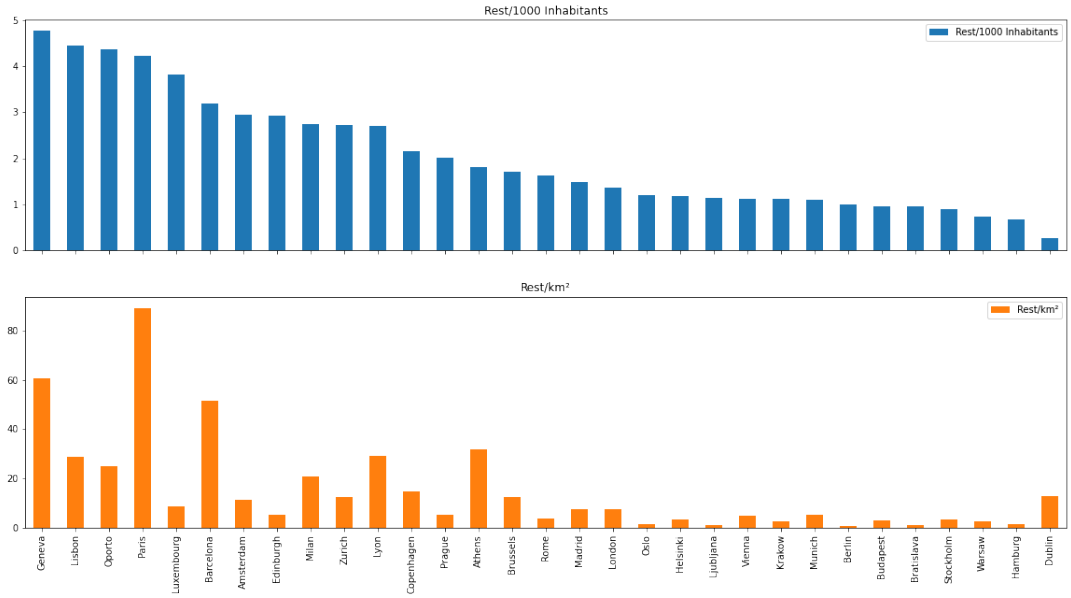

Figure 2. Number of restaurants per inhabitant and per square kilometer.

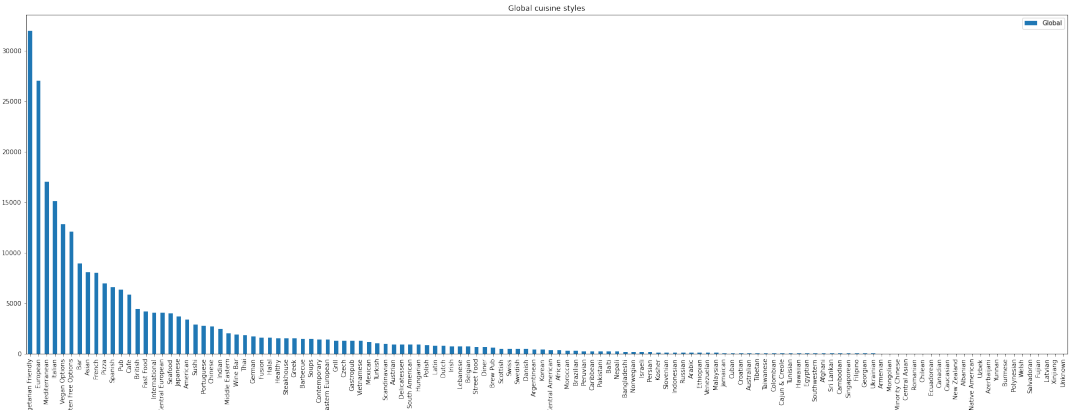

Figure 3. Global cuisine styles in all cities.

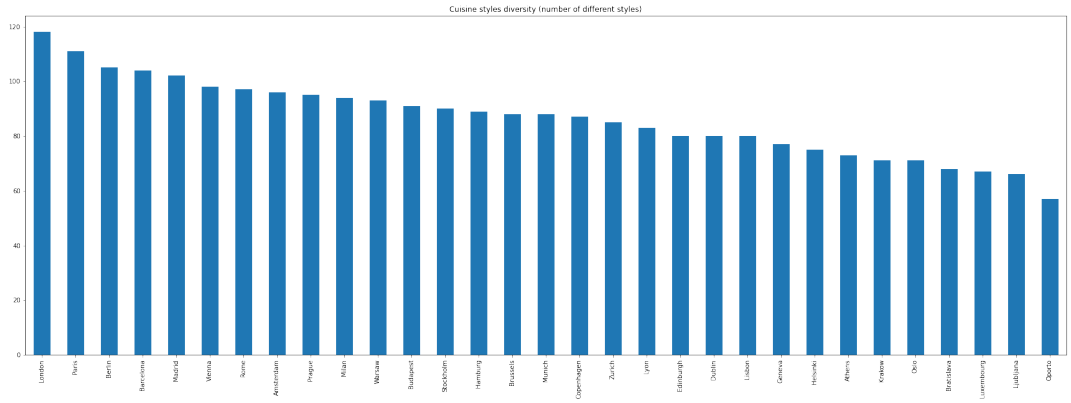

Figure 4. Cuisine styles diversity in each city.

Figure 5. Special diets ratio for each city.

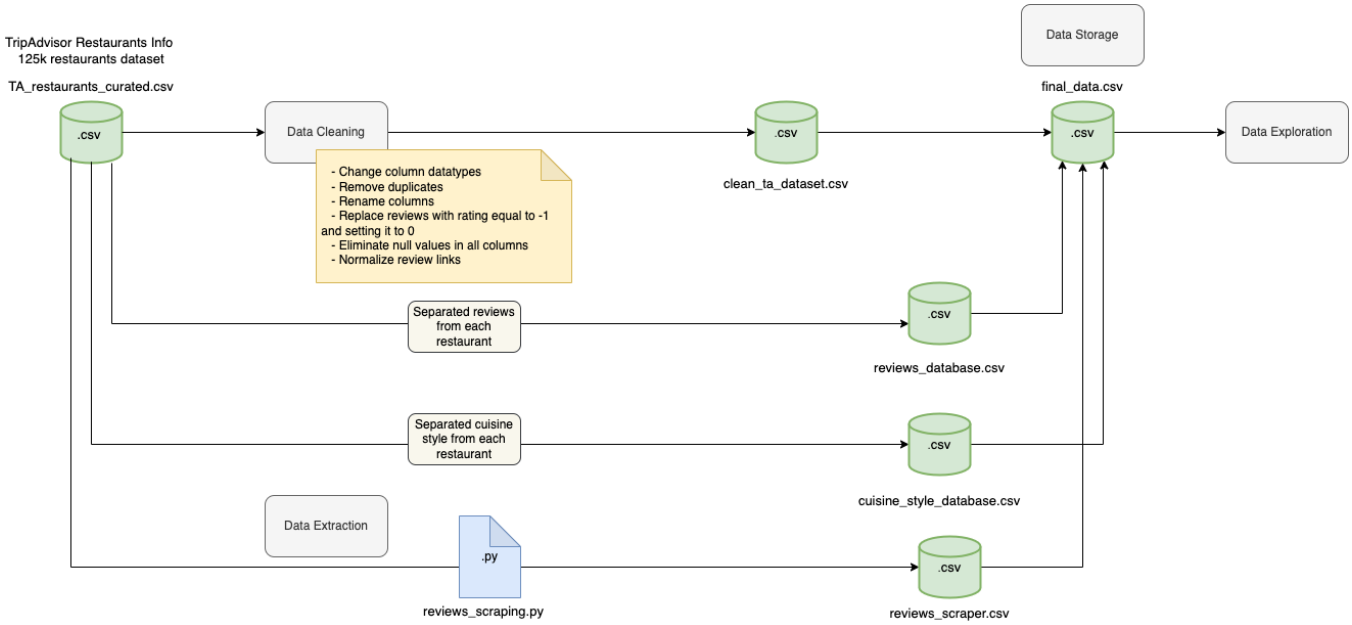

Figure 6. Data processing pipeline.

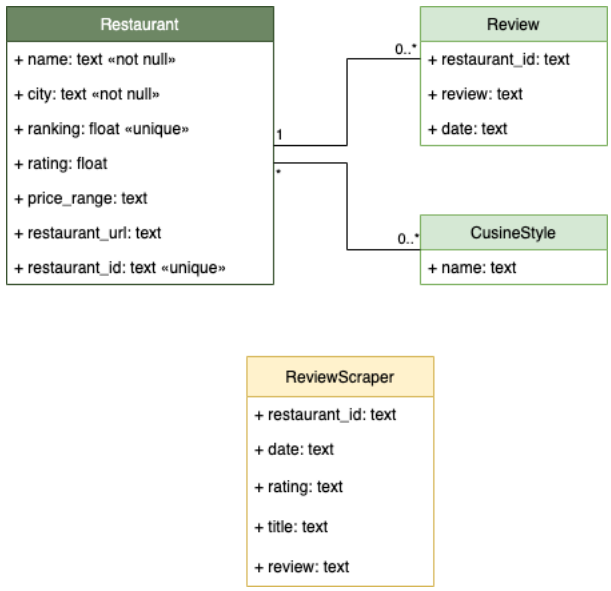

Figure 7. Conceptual model.

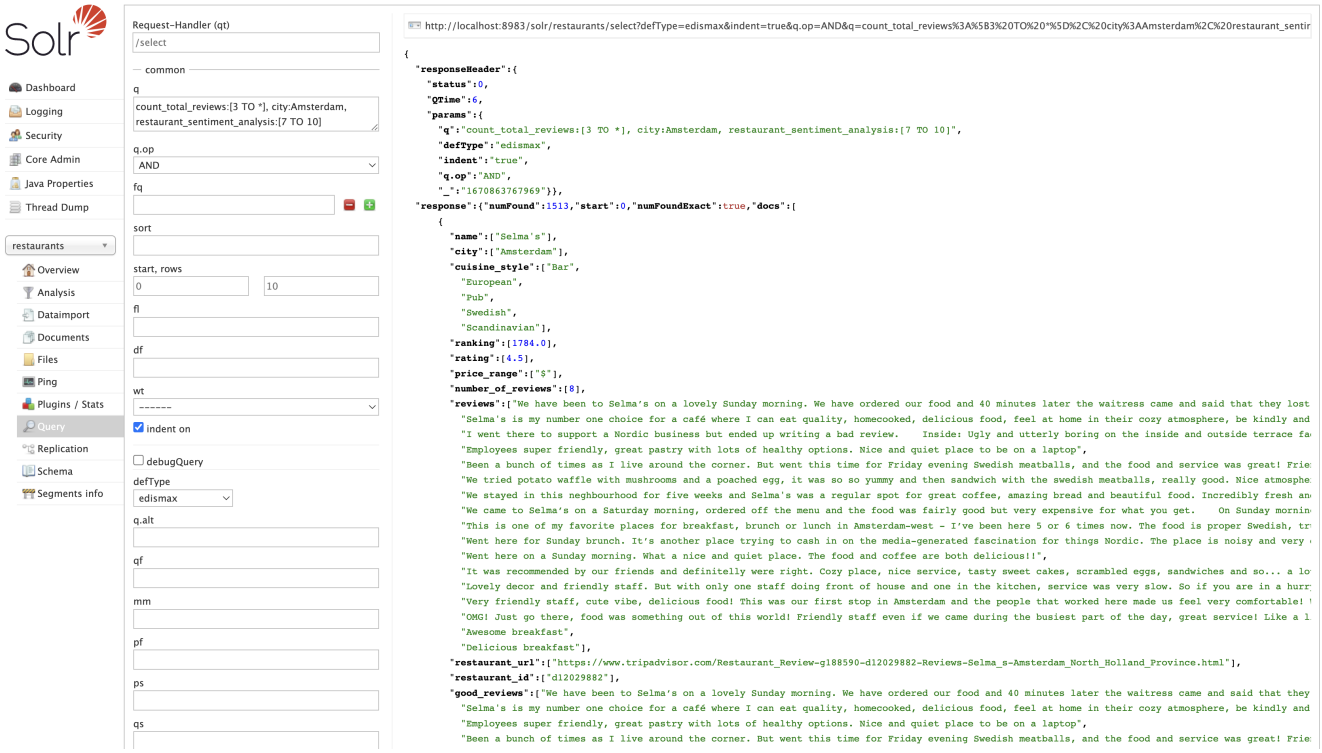

Figure 8. Basic search.

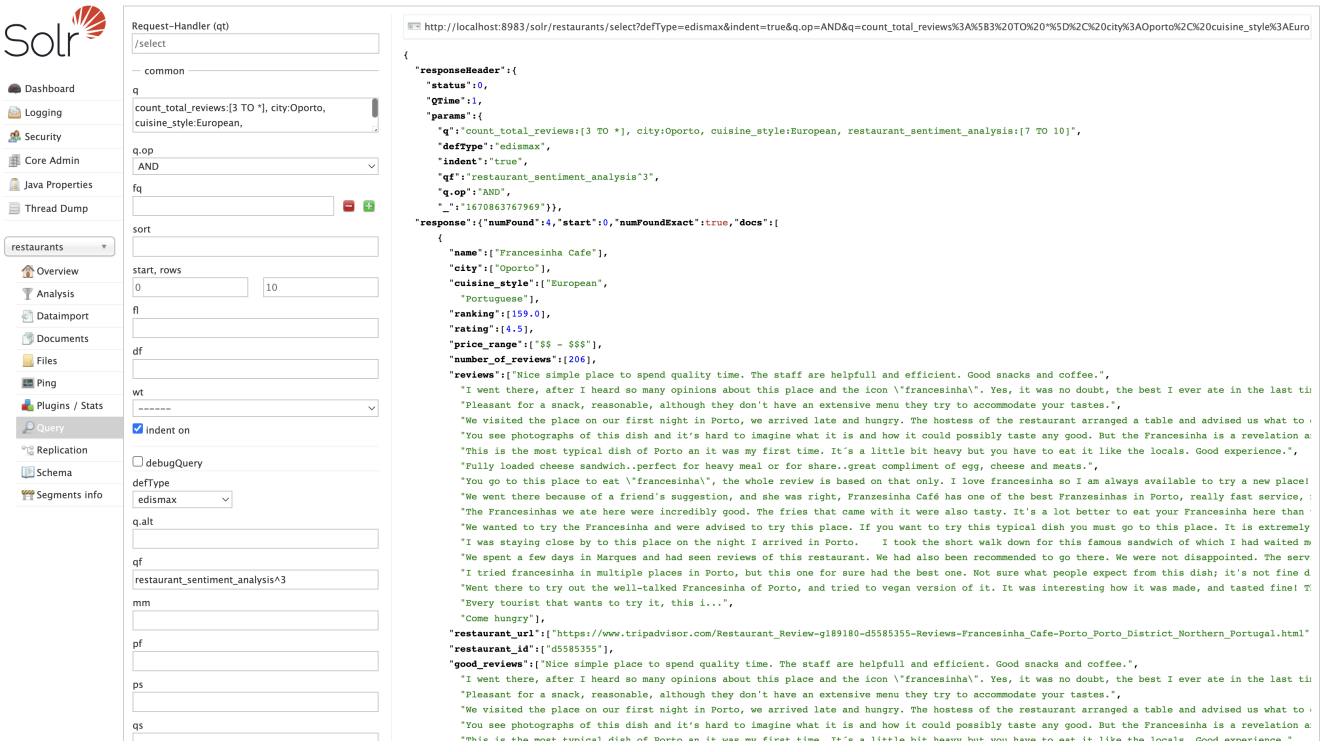

Figure 9. Weighted search.

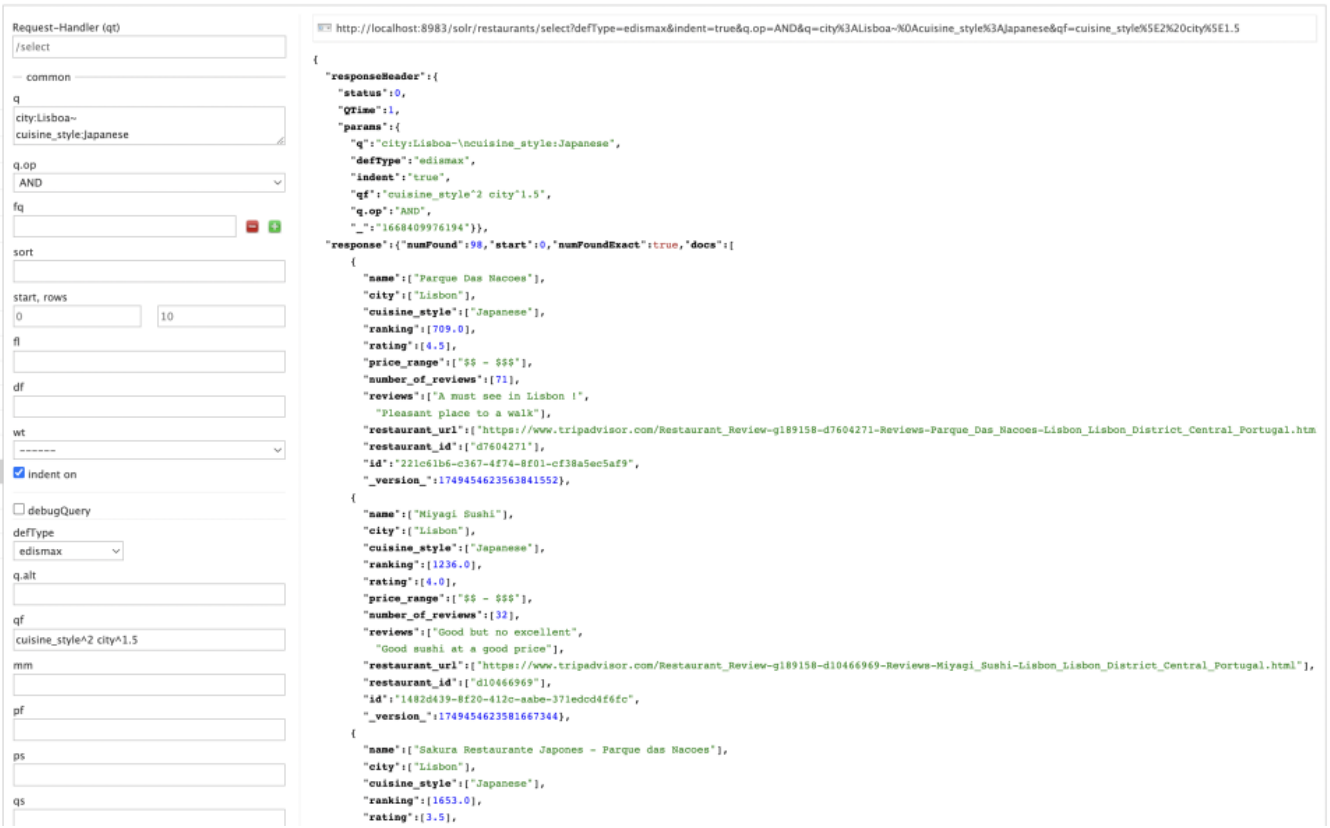

Figure 10. Fuzziness search.

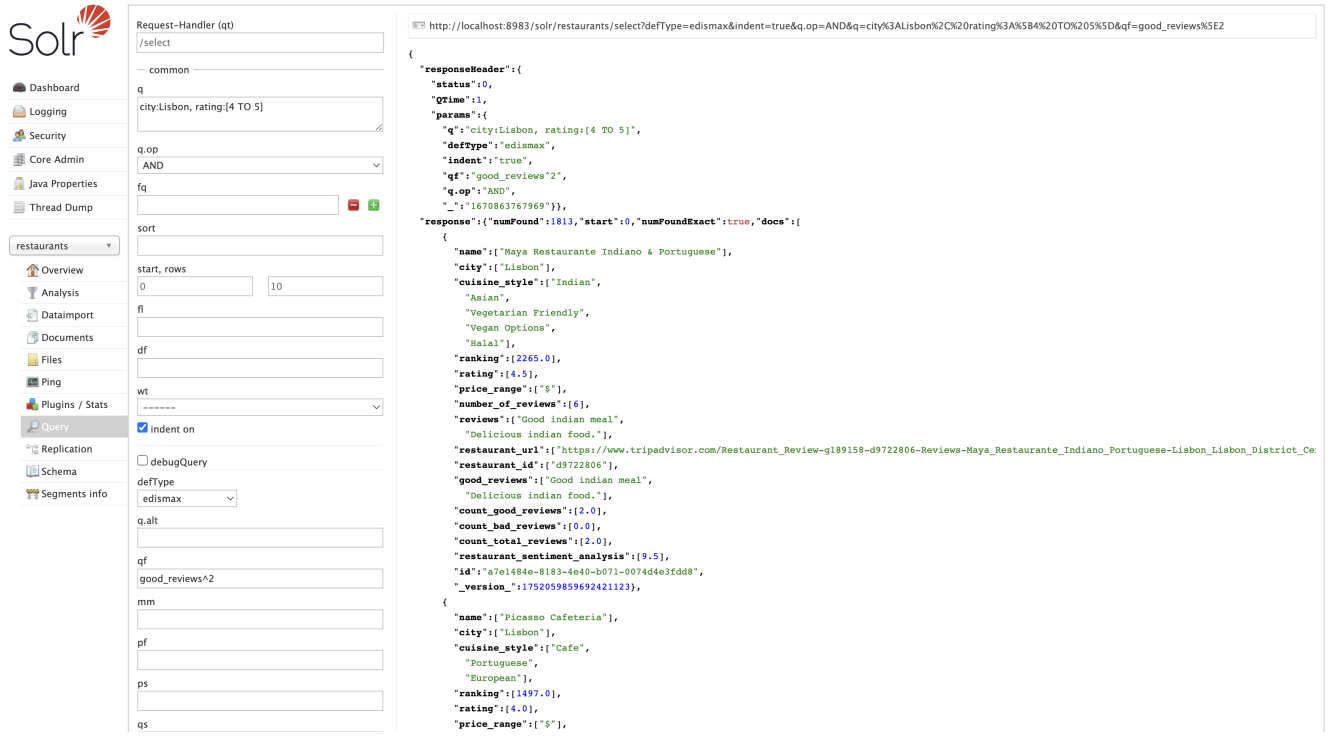

Figure 11. Sentiment Analysis of reviews is used to search.

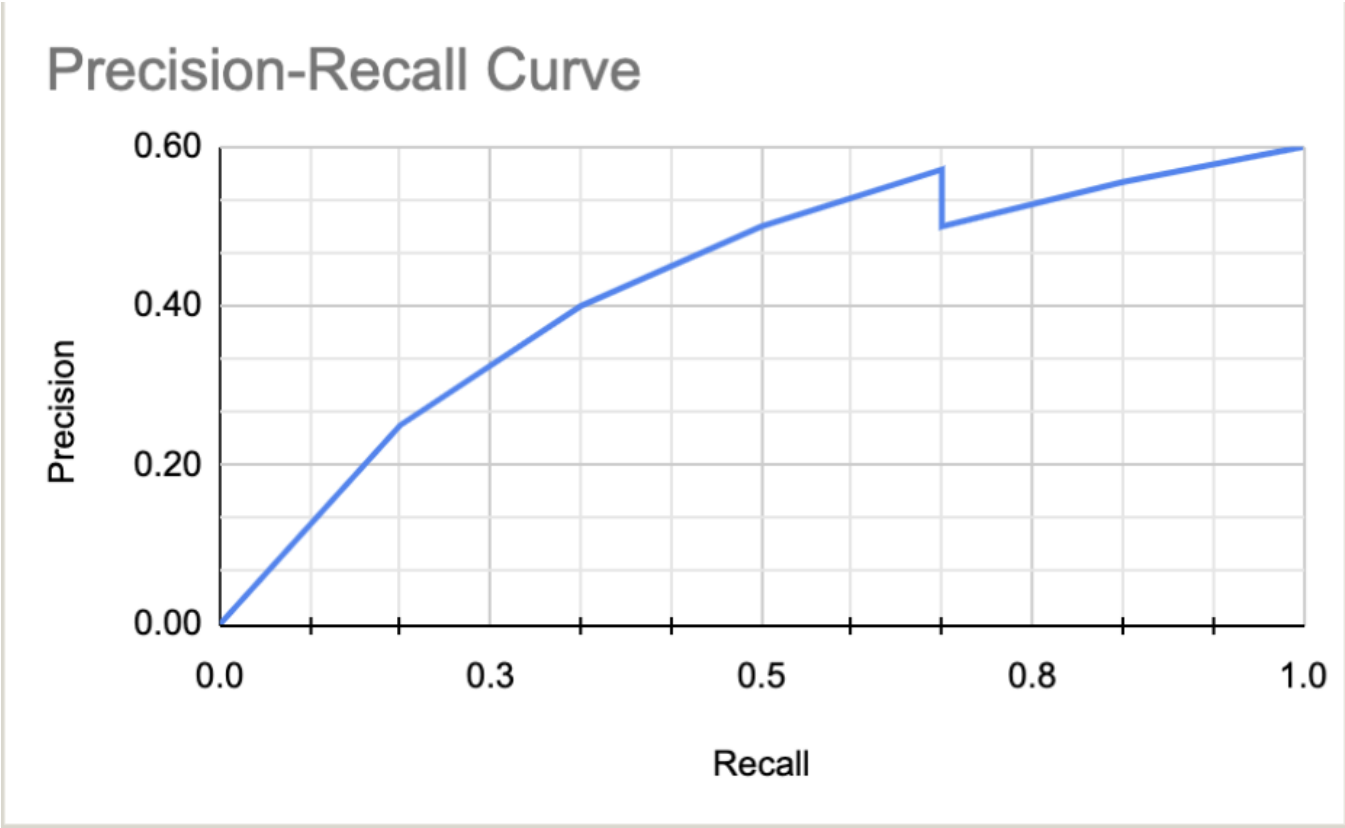

Figure 12. Precision Recall graph for Query 1 with default weights.

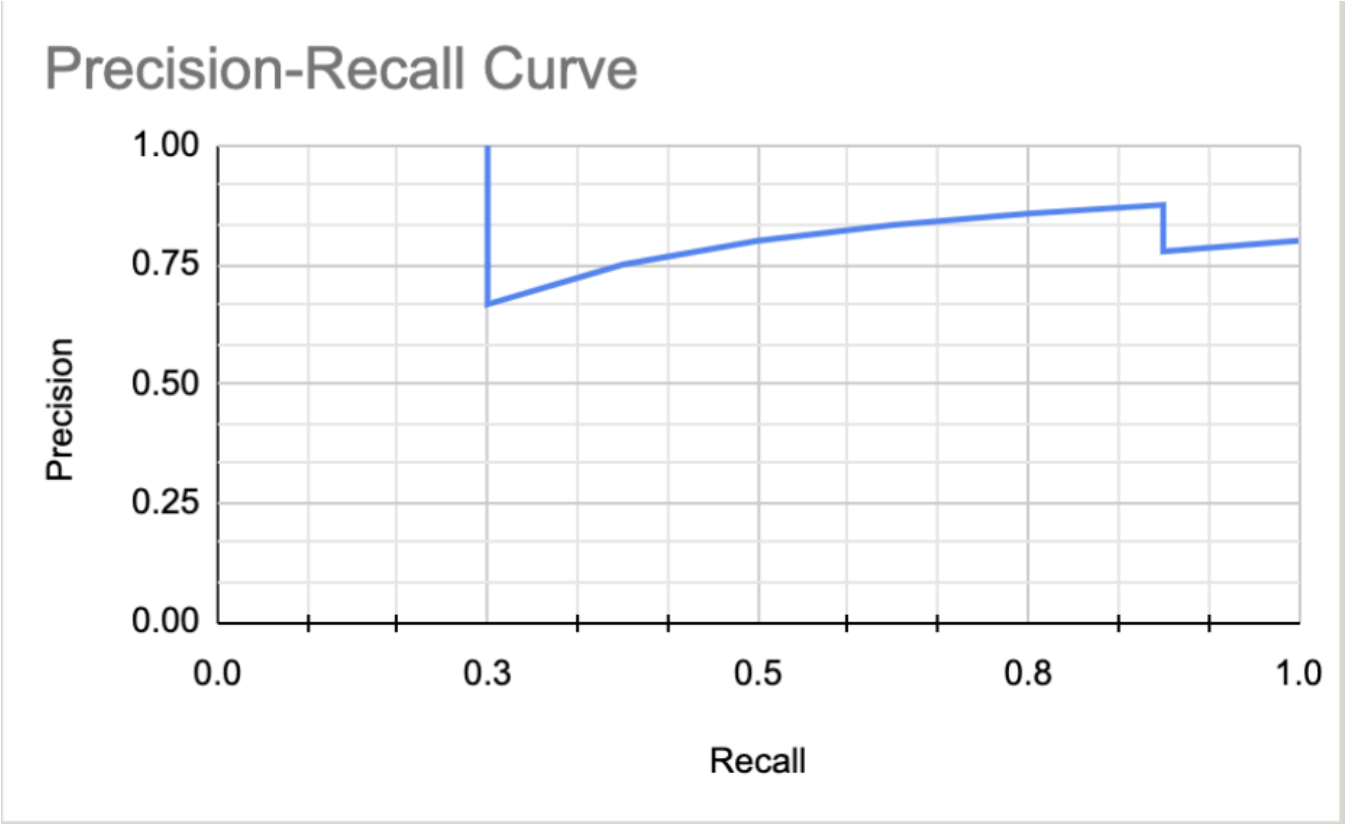

Figure 13. Precision Recall graph for Query 1 with weighted weights.

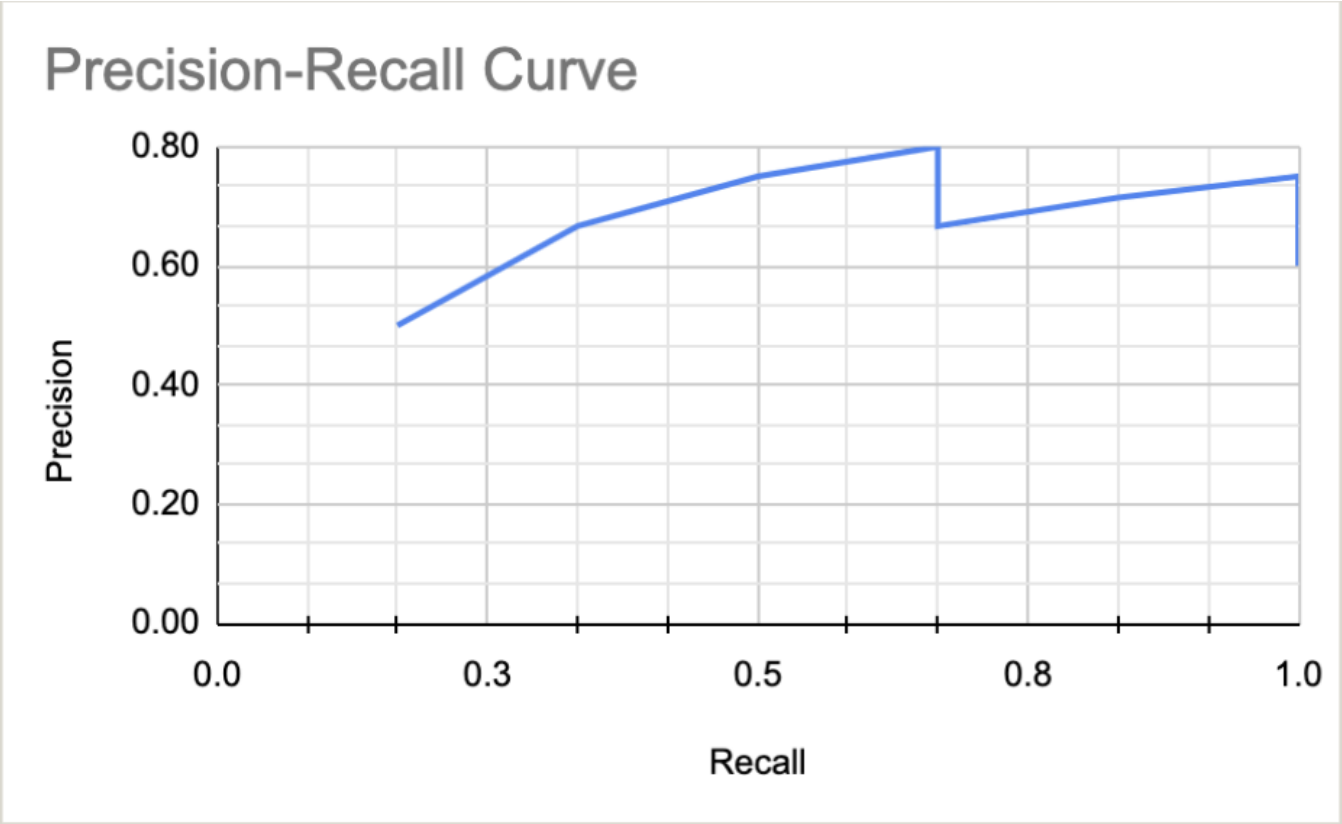

Figure 14. Precision Recall graph for Query 2 with default weights.

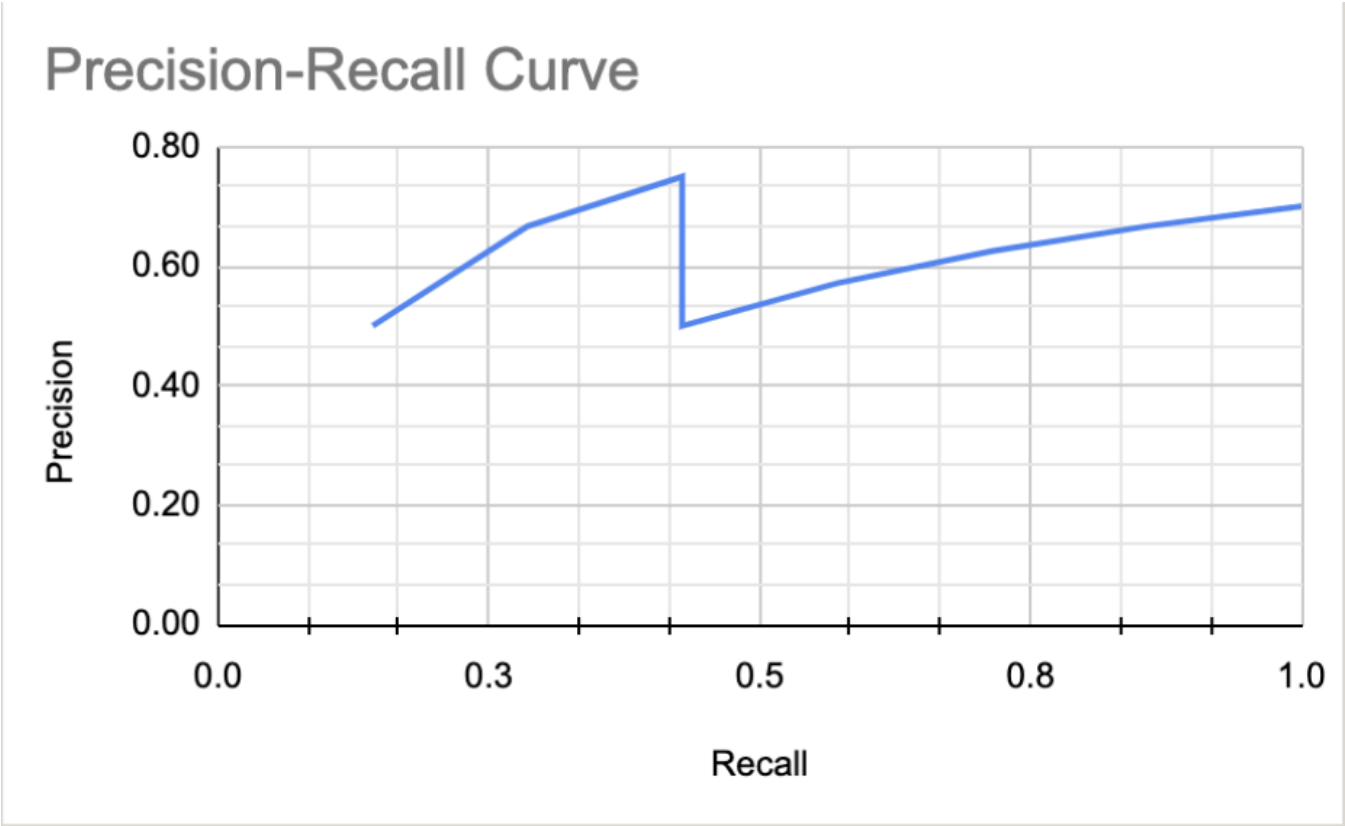

Figure 15. Precision Recall graph for Query 2 with weighted weights.

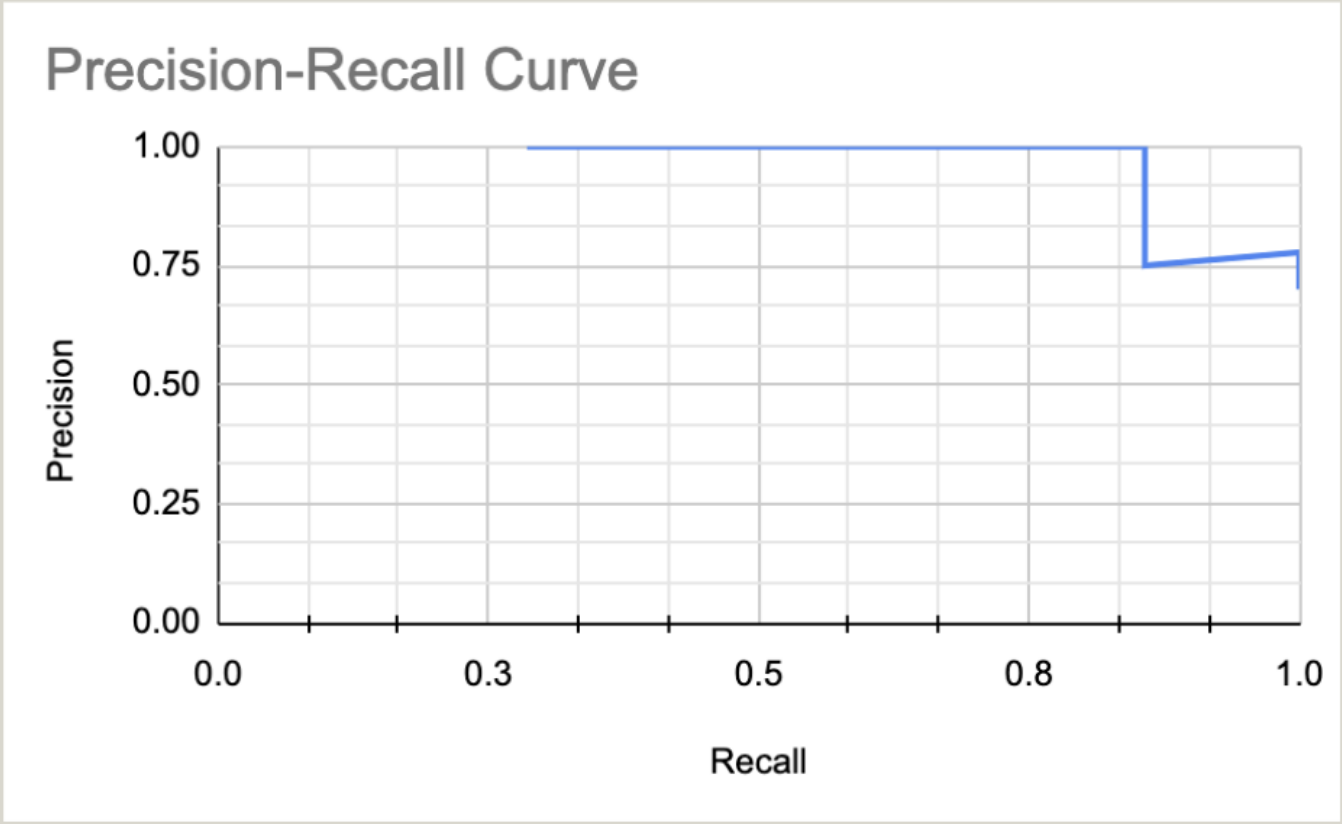

Figure 16. Precision Recall graph for Query 3 with default weights.

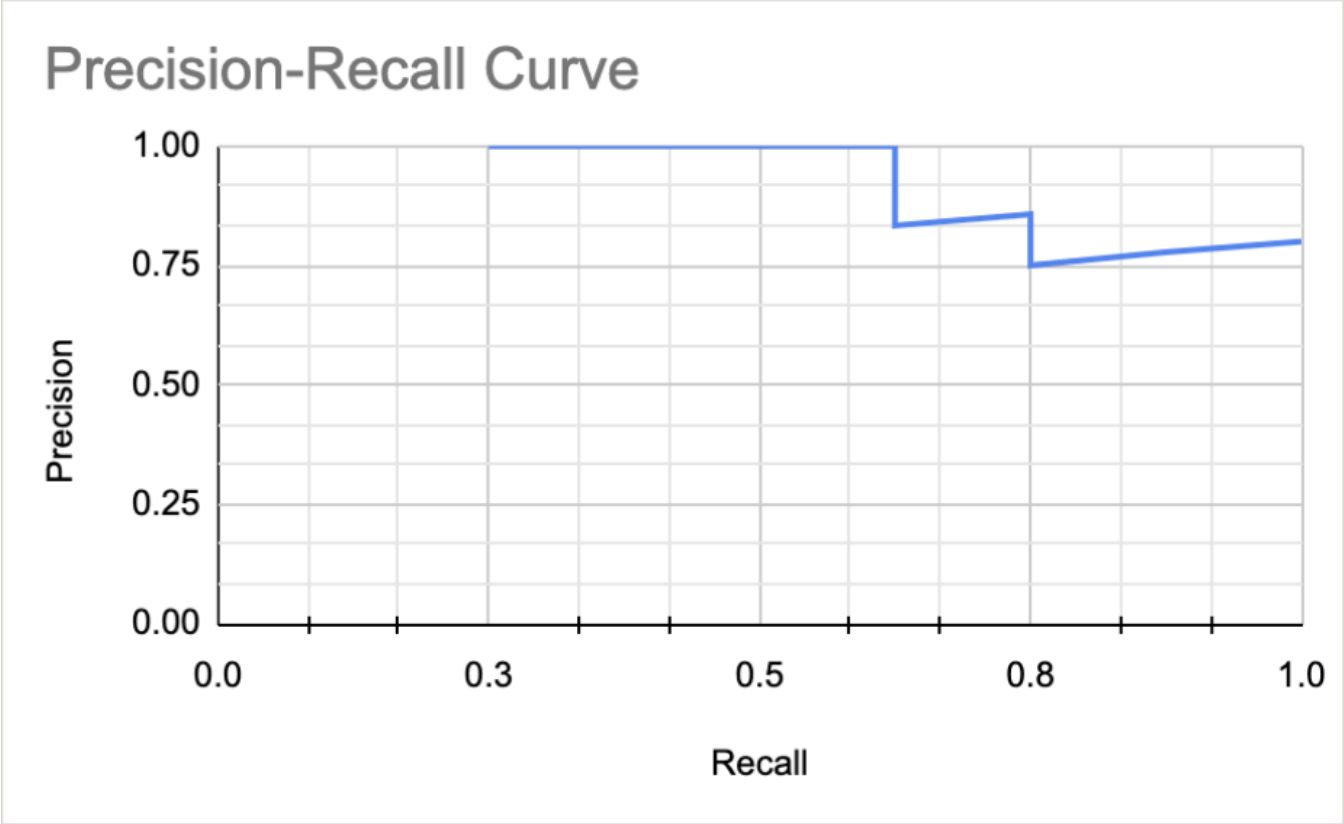

Figure 17. Precision Recall graph for Query 3 with weighted weights.

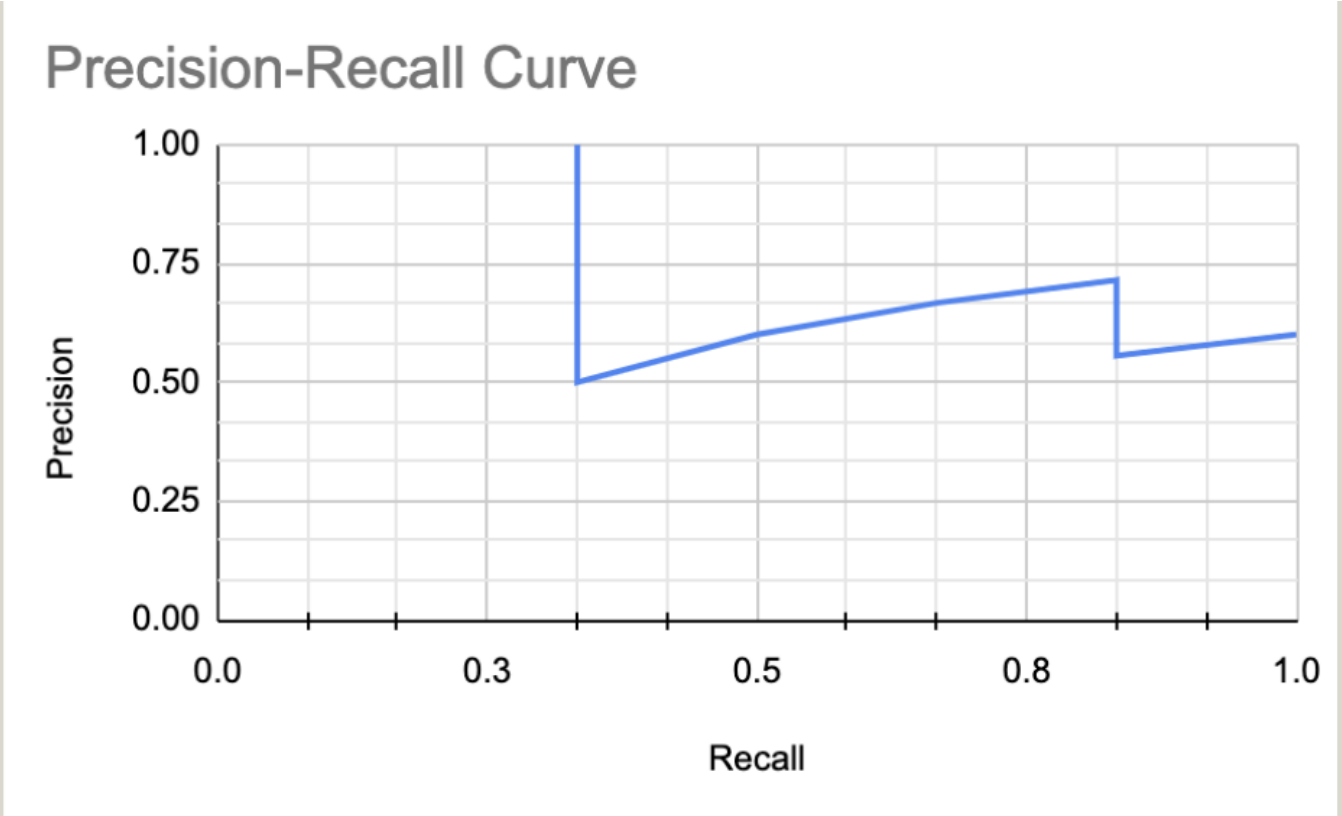

Figure 18. Precision Recall graph for Query 4 with default weights.

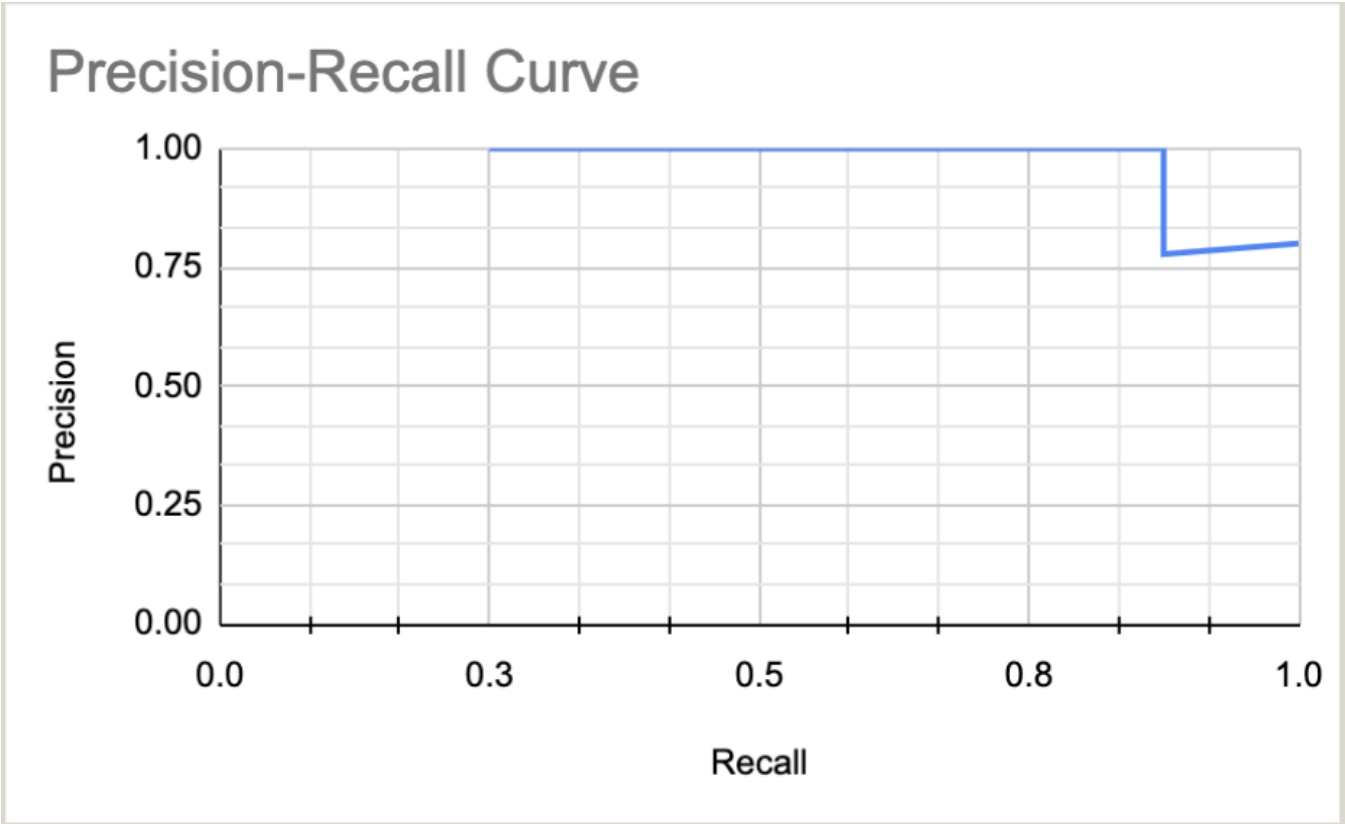

Figure 19. Precision Recall graph for Query 4 with weighted weights.