Privago: Hotels Search System based on their Reviews

INFORMATION PROCESSING AND RETRIEVAL – PRI
MILESTONE #2: INFORMATION RETRIEVAL

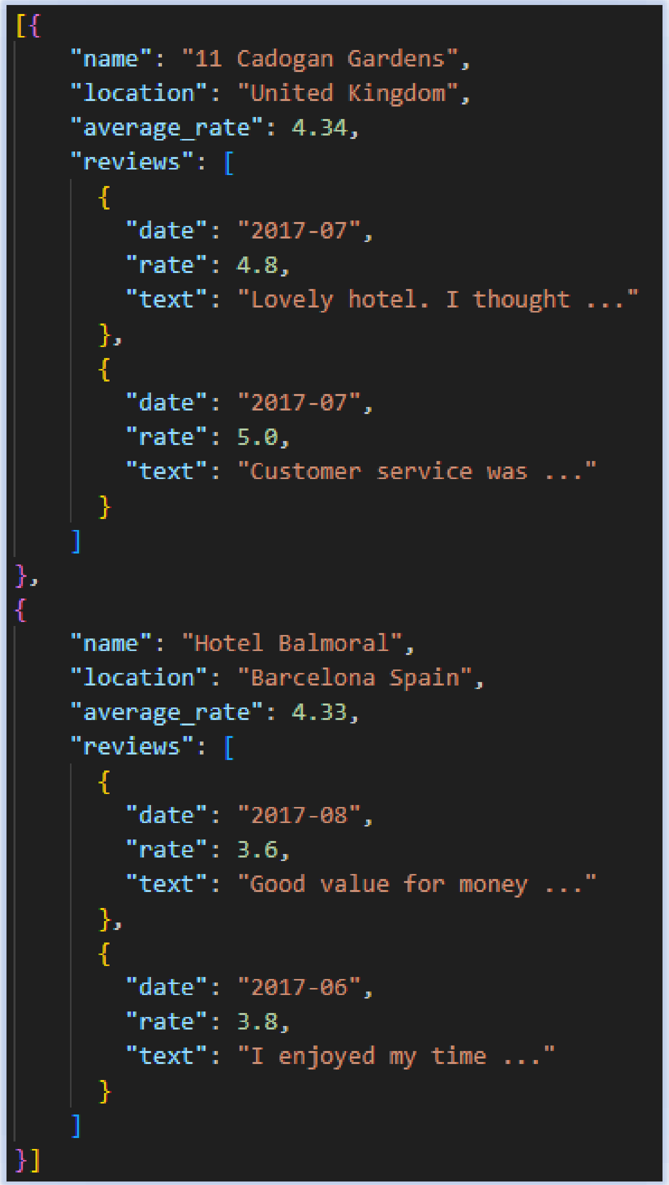
André Ávila – up202006767@up.pt
André Costa – up201905916@up.pt
Fábio Morais – up202008052@up.pt
Fábio Sá – up202007658@up.pt

Milestone 2

- Milestone 1 Overview
- Documents
- Configuration
- Indexing
- Retrieval
- Evaluation
- Conclusions on the topic

Milestone 1 Overview

- The data used for the ***Information Retrieval*** phase was extracted from Kaggle website regarding hotels and their reviews from 4 different datasets. Following the completion of the data preparation phase, the collected information was consolidated into a single json file.



```
[{
    "name": "11 Cadogan Gardens",
    "location": "United Kingdom",
    "average_rate": 4.34,
    "reviews": [
      {
        "date": "2017-07",
        "rate": 4.8,
        "text": "Lovely hotel. I thought ..."
      },
      {
        "date": "2017-07",
        "rate": 5.0,
        "text": "Customer service was ..."
      }
    ]
},
{
    "name": "Hotel Balmoral",
    "location": "Barcelona Spain",
    "average_rate": 4.33,
    "reviews": [
      {
        "date": "2017-08",
        "rate": 3.6,
        "text": "Good value for money ..."
      },
      {
        "date": "2017-06",
        "rate": 3.8,
        "text": "I enjoyed my time ..."
      }
    ]
}]
```

Documents

The final dataset is structured as a nested document, featuring individual hotels with their attributes and an internal list of child documents, their respective reviews.

- Hotels serve as the primary display document.
- Reviews take precedence as the primary search document.

Configuration

- Given the presence of nested documents, an additional flag, ***“-format solr”***, was necessary for indexing the document into Solr. This approach yielded results with hotels and reviews treated as separate documents, with the reviews indexed under their corresponding 'parent' hotel.

- **Two distinct schemas** were implemented, each employing different index and query analyzers. The first schema followed a simple or **Solr's default approach**, while the second schema adopted a personalized and enhanced methodology referred to as the **boosted approach**.

Indexing

Simple Schema

Field	Type	Index ed
average_rate, rate	pdoubles	yes
date	string	yes
name, location, text	text_general	yes

Indexing

Boosted Schema

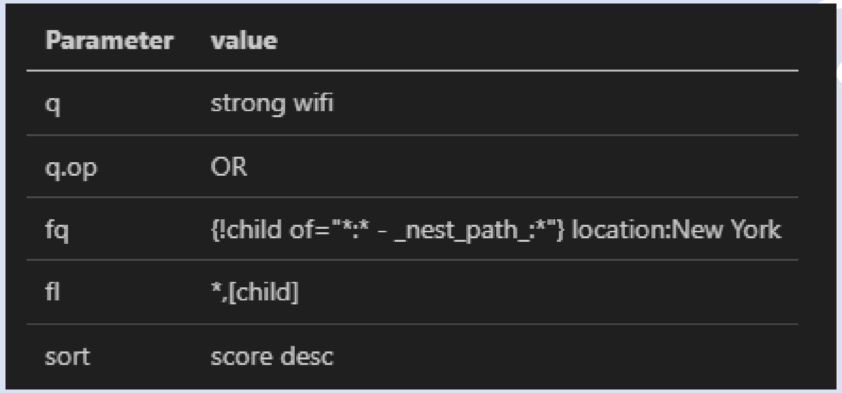
The **boosted_text** field type uses as filters for index and queries the *ASCII*, *LowerCase*, *SynonymGraph* and *EnglishMinimal* Factories.

The **SynonymGraph** is based our plain text synonyms file generated by a script using *wordnet* from *nltk*.

Field	Type	Indexed
average_rate	pdoubles	yes
rate	pdoubles	no
date	string	no
name, location, text	boosted_text	yes

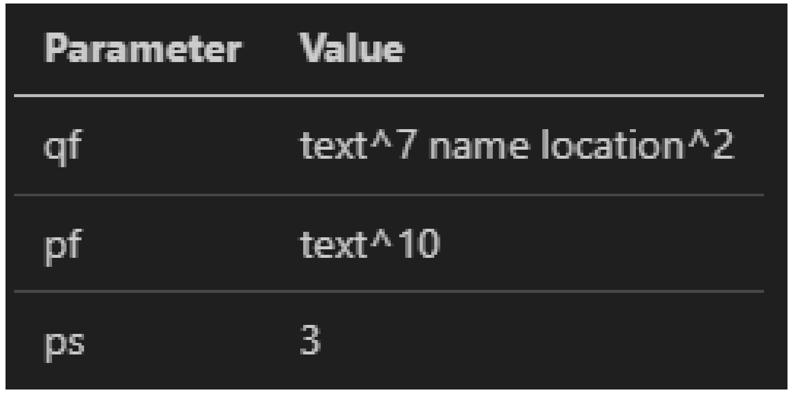
Retrieval

For query parameters used by both schemas, the system consolidates the following:



Parameter	value
q	strong wifi
q.op	OR
fq	{!child of="*:* - _nest_path_:*"} location:New York
fl	*,[child]
sort	score desc

Extra query parameters for the boosted schema:



Parameter	Value
qf	text^7 name location^2
pf	text^10
ps	3

- The eDismax parameter '**bq**' (boost query) was also explored in the approach but not included for system analysis.
- In the upcoming slides, diverse user scenarios are presented as queries, accompanied by their respective results and statistics based on precision and recall metrics.

Center of London

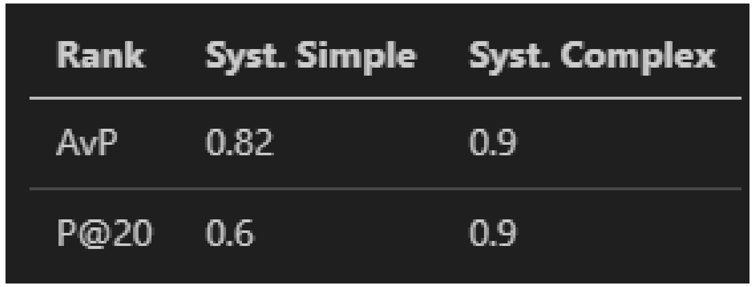
- **Information Need**: The best hotels near center of London.
- **Relevance Judgement**: In this task, the objective is to find hotels near the center of London with the highest ratings. Given the limited entries explicitly labeled as being in London, the location is set to the United Kingdom. The search is conducted using keywords like "center London" within the review text, and the results are sorted in descending order based on their rating.

- **Query**:

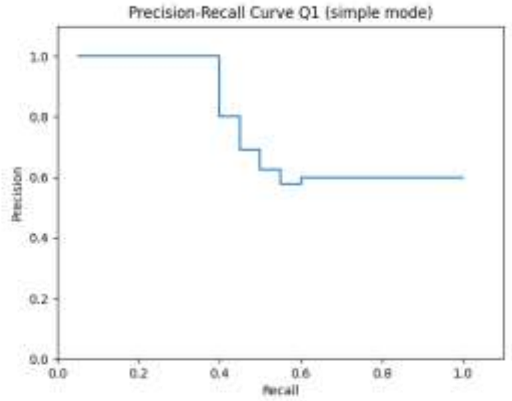
q	**(center London)**
q.op	AND
fq	{!child of="*:* -_nest_path_:*"}location:"united kingdom"
fl	*,[child]
sort	average_rate desc, score desc

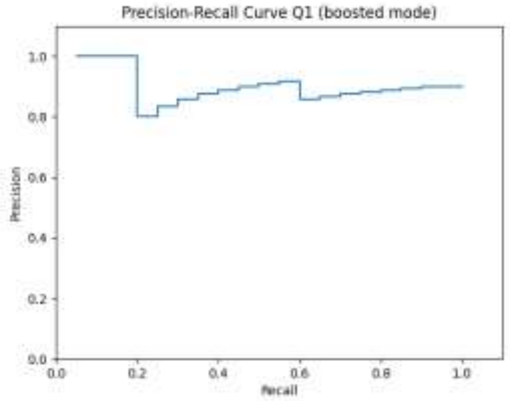
Center of London

- **Result Analysis**: Both systems did well, although there is a notable increased precision on the boosted system. The utilization of eDismax, in this query, proved to be important for the results since the 2 words "center London", when putted together, are very correlated with each other.



Rank	Syst. Simple	Syst. Complex
AvP	0.82	0.9
P@20	0.6	0.9





Breakfast or Room Service

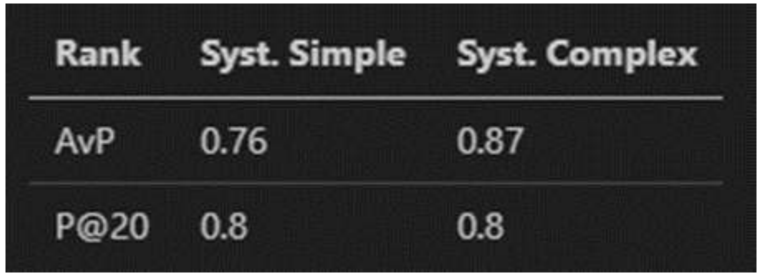
- **Information Need**: Hotels with good breakfast or good room service in New Delhi.
- **Relevance Judgement**: In this information need its intended to search for hotels with a good breakfast or a good room service in New Delhi. Therefore, the words "good breakfast" or "good room service" should appear in the same query/text of review and the location should be a filter query of the parents' documents.

- **Query**:

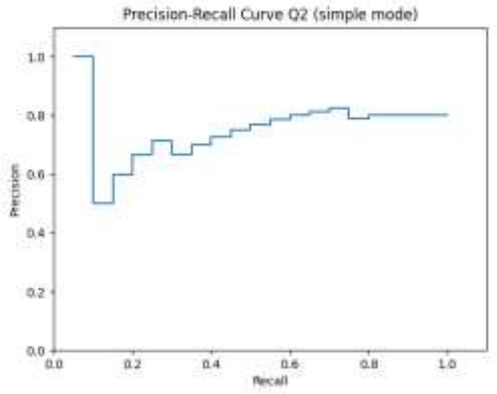
q	**(good breakfast) OR (good room service)**
q.op	AND
fq	{!child of="*:* -_nest_path_:*"}location:“new delhi”
fl	*,[child]
sort	score desc

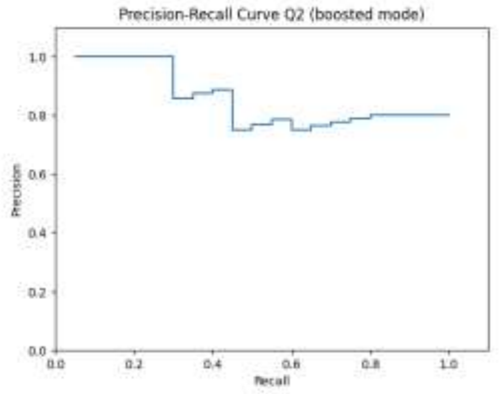
Breakfast or Room Service

- **Result Analysis**: The two systems have similar average precision. Since it is a very simple query, it is expected for good results from both systems, and it is normal for the improved one to fail in some sentences since it's using the 'ps' parameter (which is equal for every query) that allows for tokens between searched words. This would be resolved with contextual analysis referred in the "Future Work“.



Rank	Syst. Simple	Syst. Complex
AvP	0.76	0.87
P@20	0.8	0.8





Convenience and Accessibility

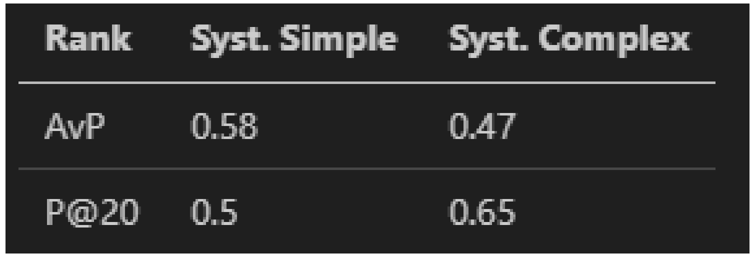
- **Information Need**: Hotel in the United Kingdom with good location and either elevator or good accessibility.
- **Relevance Judgement**: With this query we intended to gather the hotels situated in the United Kingdom which have a good location with either an elevator or good accessibility, a query for someone with reduced mobility that wants to visit the United Kingdom.

- **Query**:

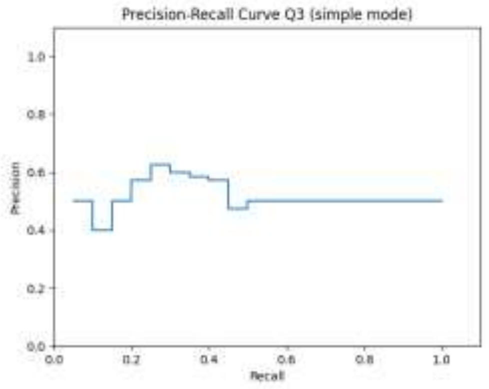
q	**good location ((elevator) OR (accessibility))**
q.op	AND
fq	{!child of="*:* -_nest_path_:*"}location:"united kigdom"
fl	*,[child]
sort	score desc

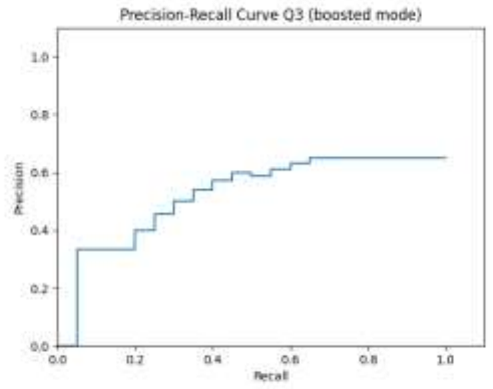
Convenience and Accessibility

- **Result Analysis**: The systems differ in average performance, with the simple one overall performing better. The reason for the complex system to score lower in the AvP (average performance) parameter but higher in the P@20 (precision) parameter, however, is due to the system not taking the context into consideration, therefore, in a query that specifies the need for an elevator, the system gathers reviews that mention the absence of one.



Rank	Syst. Simple	Syst. Complex
AvP	0.58	0.47
P@20	0.5	0.65





Vegetarian/ Vegan

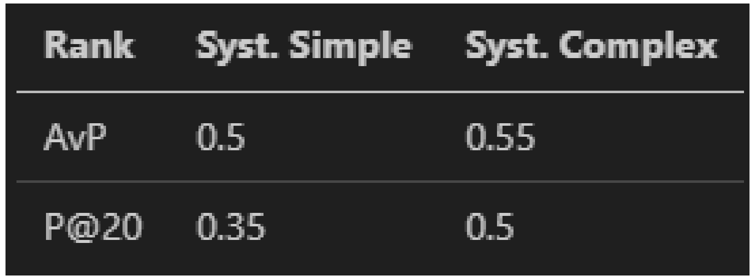
- **Information Need**: Hotels with good vegetarian/vegan options.
- **Relevance Judgement**: In this task, the objective is to find hotels with good vegetarian or vegan options. So, the words "good vegetarian" or "good vegan" should appear in the review's text. The location isn't specified.

- **Query**:

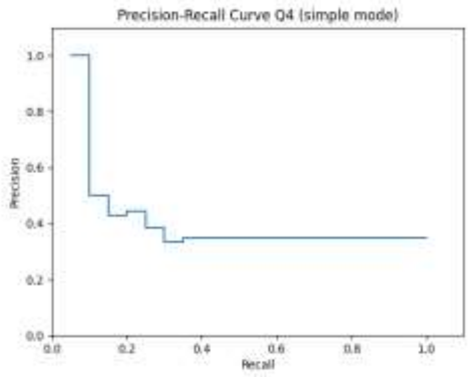
q	**(good vegetarian) OR (good vegan)**
q.op	AND
fq	{!child of="*:* -_nest_path_:*"}location:*
fl	*,[child]
sort	score desc

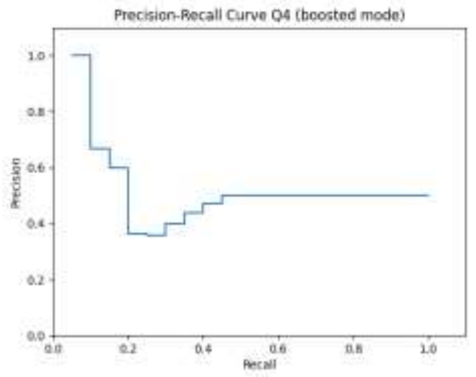
Vegetarian/Vegan

- **Result Analysis**: Both systems exhibit similar average precision values, which fall below the anticipated values for a simple query. This can be attributed to the same issue discussed in Q2. In fact, certain review texts were expressed in a negational form or contained nouns such as "lack," thereby altering the entire meaning of the sentence.



Rank	Syst. Simple	Syst. Complex
AvP	0.5	0.55
P@20	0.35	0.5





Global Evaluation and Analysis

- Considering all the results from the multiple information needs across queries, its presented in the following table the **Mean Average Precision** for both systems:

Global	System Simple	Sytem Boosted
MAP	0.665	0.6975

Conclusion: Milestone Achievements & Future Work

We have successfully accomplished all tasks set for this milestone.

The most challenging aspect of this Milestone was some of the Solr functionalities around indexing and searching on nested documents.

The Stop Words filter can be applied to boosted_text to reduce sensitivity to common words.

Sentimental and contextual analysis is relevant, given that the main source of information for the system is reviews, which inherently carry subjective connotations.

References

[Information Retrieval] (https://link.springer.com/book/10.1007/978-3-319-93935-3), 2023/10/23

[Apache Solr] (https://solr.apache.org/guide/6_6/introduction-to-solr-indexing.html), 2023/10/23

[Solr Tokenizers] (https://solr.apache.org/guide/solr/latest/indexing-guide/tokenizers.html), 2023/11/02

[Sorl Filters] (https://solr.apache.org/guide/solr/latest/indexing-guide/filters.html), 2023/11/02

[eDismax] (https://solr.apache.org/guide/7_7/the-extended-dismax-query-parser.html), 2023/11/04

[Precision-Recall Area Under the Curve] (https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html), 2023/11/07

[Solr Stop Filter] (https://solr.apache.org/guide/solr/latest/indexing-guide/filters.html#managed-stop-filter), 2023/11/07

[Indexing Nested Documents] (https://solr.apache.org/guide/solr/latest/indexing-guide/indexing-nested-documents.html#schema-configuration), 2023/11/11

[Queries in Nested Documents] (https://solr.apache.org/guide/solr/latest/query-guide/searching-nested-documents.html), 2023/11/11