Privago: Hotels Search System based on their reviews

André Costa
up201905916@edu.fe.up.pt
Faculty of Engineering - University of Porto
Porto, Portugal

André Ávila
up202006767@edu.fe.up.pt
Faculty of Engineering - University of Porto
Porto, Portugal

Fábio Sá
up202007658@edu.fe.up.pt
Faculty of Engineering - University of Porto
Porto, Portugal

Fábio Morais
up202008052@edu.fe.up.pt
Faculty of Engineering - University of Porto
Porto, Portugal

ABSTRACT

The internet's rapid growth demands appropriate systems for harnessing and connecting this massive information resource. This project addresses this need by focusing on the hotel industry, where reviews play a crucial role in shaping consumer choices. This article aims to provide a clear and well-documented explanation of the work needed in developing a robust search engine for hotel reviews. To accomplish this, data is collected from four different Kaggle datasets, cleaned and prepared, and in-depth data analysis is undertaken, as we seek to fulfill prospective search tasks such as finding hotels near the airport or finding hotels with a helpful staff. The data is then transformed into a collection on Apache Solr's search engine, where it undergoes analysis and refinement through manipulation of various characteristics.

The project achieved significant milestones. The 'Data Preparation' phase successfully navigated challenges, creating a finalized dataset that balances data richness and relevance. The 'Information Retrieval' phase executed planned tasks, overcoming challenges in handling nested documents. Evaluation confirmed the system's stability and capability to address various information needs. The 'Information Retrieval Improvement' phase concluded, implementing targets such as applying the Stop Words filter for common word sensitivity and testing sentimental semantic analysis. Additionally, user interface enhancements have been successfully introduced, providing a more intuitive and user-friendly experience. This refined hotel search engine now empowers travelers with improved tools to filter accommodations based on preferences identified during the analysis phase.

CCS CONCEPTS

• **Information systems** → Information retrieval query processing.

G53, December-10, 2023, Porto, Portugal

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00

KEYWORDS

Hotels, Reviews, Information, Dataset, Data Retrieval, Data Preparation, Data Analysis, Data Processing, Data Refinement, Pipeline, Data Domain Conceptual Model, Information Retrieval

ACM Reference Format:
André Costa, André Ávila, Fábio Sá, and Fábio Morais. 2023. Privago: Hotels Search System based on their reviews. In *Proceedings of PRI (G53).* ACM, New York, NY, USA, 18 pages.

1 INTRODUCTION

The choice of the hotel reviews theme is motivated by the large amount of information's sources and the rich diversity of attributes it encompasses. Hotel reviews, as a research focus, hold substantial importance in the modern information landscape. They not only provide valuable insights into the hospitality industry but also serve as a prime example of data diversity, combining numerical rates, submission dates, and personal, subjective narratives. This diversity introduces intricacies in data structuring and presents challenges in contextual search, making it an ideal choice for aligning the search system with real-world scenarios. Thus emphasizing practical applicability and the development of robust information retrieval solutions.

This document is structured into several major sections. We commence with **Data Extraction and Enrichment**, where we introduce the data sources, briefly characterize the datasets, and assess data quality. Subsequently, **Data Preparation** outlines the selection criteria, processing methods, and data storage procedures for hotel-related information and associated reviews, following a clear and reproducible pipeline.

In **Data Characterization** we delve into the evaluation and visualization of the refined data. This involves examining various criteria and relationships, from the Domain Conceptual Model to Word Clouds.

Possible Search Tasks provide an overreaching interpretation of the results, guiding the identification of suitable research objectives for the **Information Retrieval**, where the documents are indexed and searched using Apache Solr [1] with a defined schema and defined parameters. Finally, the **Evaluation** analyses the results obtained by custom queries on the prepared search engine, which was defined on the previous section.

2 DATA EXTRACTION AND ENRICHMENT

After conducting research for relevant data in terms of variety and quantity, four datasets in CSV format from different regions were selected through the Kaggle [2] platform. Table 1 provides a characterization of the acquired datasets:

Table 1: Initial datasets characterization

Dataset	Features	Hotels	Reviews	Size(MBs)
Datafiniti's Hotel Reviews	26	1400	10000	124.45
Hotel Review Insights	7	570	7000	1.31
London Hotel Reviews	6	20	27329	22.85
Europe Hotel Reviews	17	1493	515000	238.15

The Datafiniti's Hotel Reviews [3] dataset was taken from Datafiniti's Business Database [4] through sampling. Hotel Review Insights [5] is a compilation of hotels around the world through web-scraping of reviews found on Booking.com [6]. London Hotel Reviews [7] is a sample taken and partially refined from a DataStock dataset [8]. Finally, Europe Hotel Reviews [9] also results from web scrapping of hotel reviews across Europe published on Booking.com [6].

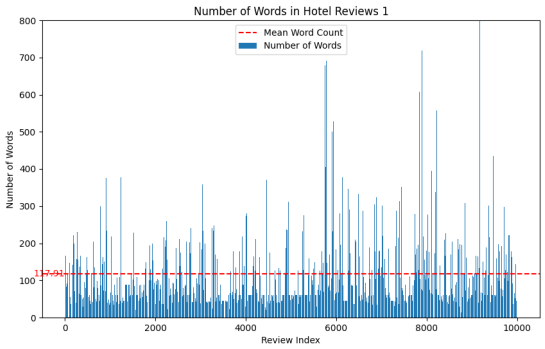

Figure 1: Average words per review in Datafiniti's Hotel Reviews dataset

All datasets have a public use license and, according to the Kaggle platform, a usability index greater than 8. This index is justified, given that the elimination of null, repeated or non-informative entries practically did not eliminate nothing.

The datasets contain common features, numerical data, such as rate, review date, and textual data, such as review text, hotel location and name. The last dataset contains two additional parameters, positive review and negative review. The features stated were extracted in this step and refined in Data Preparation.

3 DATA PREPARATION

In this section, is presented the structured data preparation pipeline [11] that was developed for the project. This pipeline embrace various data cleaning and restructuring procedures aimed at achieving a clean, uniform, and ready to analyze dataset. The goal of this stage is to build a strong basis for insightful analysis.

The data preparation process began with a comprehensive cleaning phase, where the primarly focus was the removal of records containing **empty or null values** in any attribute. Simultaneously, was identified and eliminated incomplete or uninformative data, including strings with **uninformative text**, such as "no comments available for this review.", using Python. This combined cleaning step ensured that the dataset was cleansed in detail for the normalization phase.

With the data cleaned, attention was turned to **attribute normalization**. Given the presence of diverse datasets with varying formats, comprehensive normalization process was needed. This included standardizing attributes such as **"positive_reviews"** and **"negative_reviews"** into a unified **"review_text"** attribute for the 4th dataset as is demonstrated in the Pipeline Diagram [Figure 11]. Additionally, **date formats** were normalized to ensure uniformity and suitability for analysis. The date format was set as "year-month" due to the absence of day-specific review information in the second dataset and its irrelevance to the research targets.**Rate scales** were also normalized to a common range and converted to floating-point values, facilitating comparative analysis ([0.0, 5.0]).

In addition, was established a **standardized naming** convention to address variations in **hotel names**, such as from *"45 Park Lane - Dorchester Collection"* to *"45 Park Lane Dorchester Collection"*. This step was necessary to facilitate the aggregation step and the addition of the feature "average_rate" to each hotel entity, referenced below. **Location standardization** involved reducing location names to their last two words, preserving only the capital and country names.

To gain insights into the textual content, we calculated the temporary column **word count** for each review across all datasets. This analysis was facilitated using the Pandas [10] Python tool, allowing us to extract valuable information such as quartile ranges and make informed decisions during the review deletion phase. This process enabled us to identify and manage reviews with either an insufficient word count or an excessively high word count. We achieved this by removing reviews falling below the 25% threshold (first quartile) and those exceeding the 75% threshold (fourth quartile). This step was done separately for each dataset, due to the discrepation of each average word counting [Figure 12] [Figure 13].

At this stage, all the datasets were successfully merged into a single, consolidated dataset, which streamlined the remaining preparation tasks. These tasks commenced with the computation of the temporary column **"average_rate"** for each unique hotel. This information may prove valuable for defining search criteria in future milestones.

After completing the aforementioned steps, the next phase involved determining the minimum and maximum number of **reviews per hotel** to be retained. To accomplish this, the same approach used for analyzing the number of words per review was employed, utilizing the Pandas [10] .describe() function. This statistical analysis provided essential insights into the distribution of reviews across hotels. This step was important due to the discrepation of the number of reviews per hotel [Figure 2].

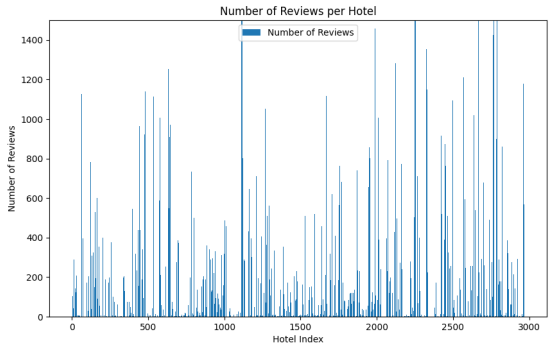

Figure 2: Number of Reviews per Hotel

First, hotels with fewer reviews, falling below the established minimum threshold (first quartile), were addressed, and they were subsequently removed from the dataset. This step was crucial in ensuring that the dataset focused exclusively on hotels with a volume of reviews contained in the second and third quartile, thus providing more meaningful insights.

Subsequently, hotels with an excessive number of reviews were taken into account. To handle this situation, a strategy was implemented, allowing the selection and retention of reviews while preserving the proportion of reviews per rate category for each specific hotel, i.e., its global rate. This approach guaranteed a balance between the "average_rate" and the rate of the selected reviews.

The final step in the data preparation phase involved organizing the data into the **desired JSON file format**, which was designed based on our UML diagram [Figure 8] and with a focus on the primary objective of the search tool. This format consisted of a collection of JSON objects, each representing a "Hotel" entity. Within each "Hotel" object, were included not only its associated attributes (name, location, average_rate) but also the related reviews, presented as JSON objects themselves. Each review has a corresponding text, rate and submission date.

4 DATA CHARACTERIZATION

In this section there is a characterization of the documents that are produced by the pipeline. The graphs and tables were obtained using the Matplotlib [11], Numpy [12] and NLTK [13] libraries.

The word cloud based on the texts of the reviews supports the search tasks of the next milestone as well as the contextual search to be implemented. As expected, the words that stand out the most are those intrinsically related to the hotel industry, such as "staff", "room", "location", "breakfast". Generally, the highlighted words have a neutral tone, although several with a very positive connotation are also detectable, such as "clean", "great", "lovely", "excellent", which can also be proven by the average rate [Figure 5] given to hotels.

Figure [4] shows the distribution of the 10 most frequent hotel locations in the system. As expected, the capitals and large cities of the main tourism countries concentrate the largest number of hotels and their reviews.

Figure 3: Reviews Word Cloud

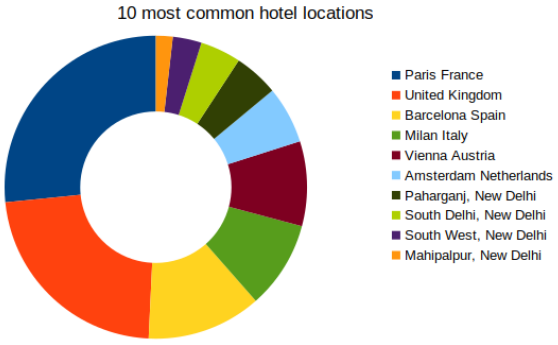

Figure 4: Hotel location distribution

As we can see in figure [5] Hotels tend to have very good reviews. The result is encouraged by their locations, as the majority are in cities with major tourist attractions and therefore with a greater cadence of reviews.

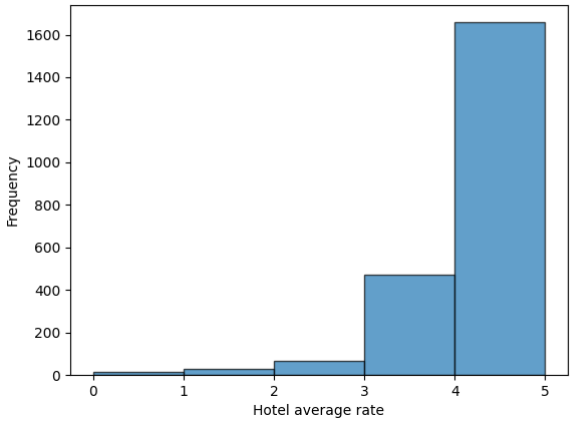

Figure 5: Average rate distribution

From the analysis of the figure [6], it can be concluded that the system includes hotels with reviews from 2010 to 2023. However, the choice of initial datasets with information relating mainly to the period 2015 and 2017 influenced their representativeness.

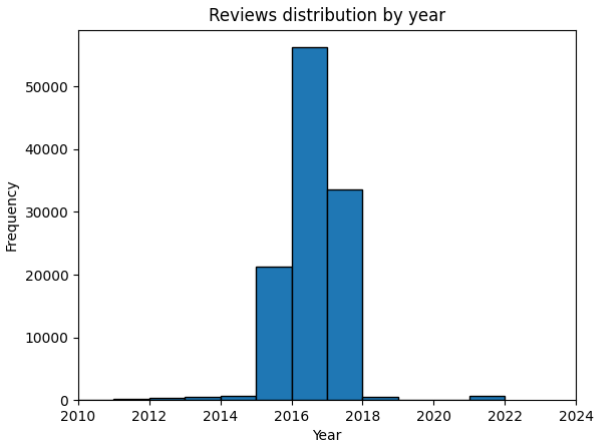

Figure 6: Reviews distribution per year

Let's look, for example, at the distribution of reviews by month in 2016 in the figure [7].

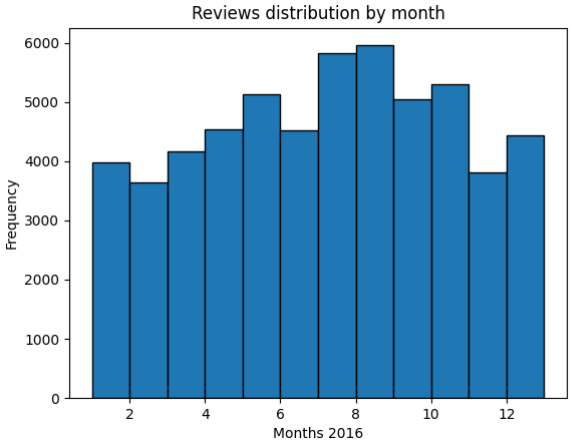

Figure 7: Month reviews distribution in year 2016

As we can see, it is July and August when the number of reviews is higher. This period corresponds to the Summer time when people normally take vacations and therefore choose to travel. This pattern is repeated for the other years under analysis.

4.1 Data Conceptual Model

After the Data preparation phase, our documents contain the following relationships:

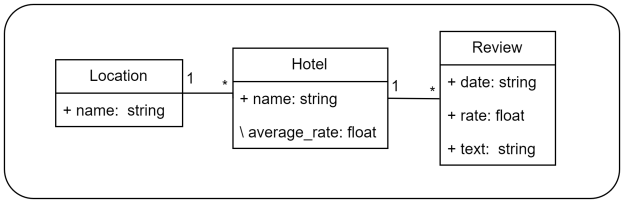

Figure 8: Data Conceptual Model

A hotel is made up of the attributes name, location and average_rate. Note that average_rate is a derived attribute that was calculated in the process based on the selected reviews. Each review has the corresponding text, rate and submission date.

Location was also considered a class because, as expected in the hotel scene, there are several hotels per region, mainly in large cities, as shown in figure [4].

5 POSSIBLE SEARCH TASKS

In the course of the data analysis journey, were unveiled valuable insights through the use of a Word Cloud Diagram [Figure 3]. This visual representation highlighted the most frequently occurring words in hotel reviews, shedding light on what matters most to travelers. Among these words, some stood out as pivotal in understanding the key factors that influence hotel choice and guest satisfaction.

"Location" emerged as one of the top considerations in travelers' decision-making processes. Whether it's proximity to local attractions, accessibility to transportation hubs, or the overall neighborhood ambiance, the location of a hotel can greatly influence the overall travel experience. Search scenarios such as **"Hotel in the United Kingdom with good location and either elevator or good accessibility."** and **"The best hotels near center of London."** can help travelers selecting accommodations that align with their preferred locations and provide convenient access to their destinations.

The words "breakfast" and "service" had a prominent presence in the word cloud [Figure 3], suggesting a keen interest in this aspects. Whether it's a hearty breakfast buffet or unique morning options, travelers seek accommodations that cater to their breakfast preferences. Another one of the foremost considerations in hotel selection is the quality of the room and its amenities. Words like "room," "bed," and "bathroom" featured prominently in our Word Cloud [Figure 3], underscoring the importance of these aspects to travelers. Searches related to "room"/"bed" quality or "bathroom" sanitation can guide travelers to accommodations that prioritize comfort and cleanliness. A search such as **"Hotels with good breakfast or good room service in New Delhi"** could be instrumental in identifying hotels that excel in the breakfast provided or in the room service available to their guests.

The context about the hotels left in their reviews adds the possibility of searching for specific details, for example, a person that follows a vegetarian diet might be struggling to find compatible hotels, this hassle could be resolved with a search such as **"Hotels with good vegetarian/vegan options"**.

6 INFORMATION RETRIEVAL

Information Retrieval [14] is the process of finding and extracting relevant information from large collections of naturally unstructured data, such as texts. This extraction is based on documents, which are the result of restructuring the initial data, and the output is sorted by relevance, becoming the main challenge.

This section presents the indexing and query methods used in this information retrieval system powered by previously constructed documents.

The implementation of the search system is based on Apache Solr [1] , an open-source tool that offers various features relevant to the project's purpose, including distributed and fast indexing, scalability, and advanced search capabilities surpassing a full-text match.

6.1 Document Characterization

The documents to be indexed and searched in the system are those resulting from the processes of data extraction, enrichment, and aggregation in the pipeline described above.

Therefore, a hotel is a document consisting of a name, average rating, location, and has a set of associated reviews. These reviews have their corresponding date, the assigned rating, and the user's comment about the hotel.

6.2 Indexing Process

Indexing serves as a fundamental step in Information Retrieval, optimizing search efficiency by organizing the data. It involves creating a structured index that enhances both search speed and scalability. Without proper indexing, search systems would face challenges, resulting in slower response times and increased computational overhead.

In Solr, various types of indexing exist for document fields and associated queries, based on a Tokenizer [15] and Filters [16]. While Tokenizers create a token stream from the original string following a predefined rule, Filters transform these tokens for consistency in subsequent searches and matches.

In this specific case, the focus was primarily on indexing textual fields, as they provide the most context and information for searches. Conversely, given the project's context, it is not expected to search for specific dates or review ratings. Therefore, these latter two document fields were not indexed.

Textual fields were indexed by instantiating a new data type. The **'boosted_text'** index analyzer includes:

- ***StandardTokenizerFactory*** tokenizer: splits texts based on punctuation and spaces;
- ***ASCIIFoldingFilterFactory*** filter: handles special characters and accents, converting them to their equivalent ASCII form;
- ***LowerCaseFilterFactory*** filter: converts all characters to their lowercase counterparts;
- ***SynonymGraphFilterFactory*** filter: expands each token to include variations based on its synonyms;
- ***EnglishMinimalStemFilterFactory*** filter: reduces each token to its root form, facilitating the search for variations of specific terms;

Fields with native values were defined using Solr's default types. The English language was chosen for both stem assignment and synonym generation, aligning with the language of the manipulated data.

The **SynonymGraphFilterFactory** is crucial in this context. Since the search is conducted based on reviews, which are inherently subjective, derived from natural language and rich in adjectives, it is important not to rely on specific terms but rather to match synonyms of terms.

The same structure was used for the query analyzer. Thus, the indexing of the final document can be characterized by the following schema:

Table 2: Schema Field Types

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

6.3 Retrieval Process and Setup

The approach implemented involves two schemas: a simple schema utilizes default field types for each field, while the more complex schema incorporates instantiated field types for enhanced search capabilities.

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

- **Query (`q`)**: Focuses on the most valuable words in the query.
- **Query Operator (`q.op`)**: Utilizes OR | AND for query operations.
- **Query Filter (`fq`)**: Defines a query that can be used to restrict the superset of documents.
- **Filter List (`fl`)**: Limits the information included in a query response.
- **Sort Field (`sort`)**: Specifies the sorting value for the results.

Table 3: Query parameters

Parameter	Value
q	(strong wifi)
q.op	AND
fq	!child of="*:* - _nest_path_:*" location: "New York"
fl	*,[child]
sort	score desc

The need of the fl and fq parameters results from the inclusion of nested documents [17] [18] . The final dataset consists of hotels, each containing a list of reviews. Recognizing the relevance of both hotels and reviews as distinct documents, a distinct approach to the search process was necessary. The combined use of fl and fq proved to be efficient in this step.

Additional query criteria the simple schema uses the default type, while the boosted schema employs the type created by eDismax [19] with specific parameters for optimizing search engine results:

- **Query Field with Optional Boost (`qf`)**: This assigns weights to specific fields in the search;
- **Phrase-Boosted Field (`pf`)**: Focuses on selecting more relevant terms from the query;
- **Phrase Boost Slope (`ps`)**: Defines the maximum number of tokens between searched words;

Table 4: defType eDismax parameters

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

Assigning diverse weights within the **qf** parameter prioritizes the significance of the **text** field, being the main field of search, followed by **location** and **name**. In the **pf** parameter, exclusive attention is given to the **text** field, serving as a dedicated phrase boost. This is complemented by the **ps** parameter set to 3, an average number of maximum tokens between the searched words.

The eDismax parameter **bq** (boost query) was also explored in the approach but not included for system analysis. Since it relies on the characteristic tokens of each query, it was found that it would introduce bias to the results while configuring a system that wouldn't be general enough for all information needs.

Being consistent with this boosted approach to every query has enhanced the system's query handling, leading to improvements in search results, as elaborated in the subsequent section.

7 EVALUATION

Evaluation is also a fundamental aspect of Information Retrieval, contingent on the target document collection and the type of information required. Understanding potential user scenarios is crucial for defining new designs and implementations based on received feedback. In this specific case, the evaluation was conducted from the perspective of effectiveness — the system's ability to find the right information — rather than efficiency, which pertains to the system's speed in retrieving information.

The use of individual and subjective metrics can introduce bias in evaluating the two previously instantiated systems. To address this, a set of distinct metrics based on **precision** and **recall**, such as **Average Precision (AvP)**, **Precision at K (P@K)**, **Precision-Recall curves**, and **Mean Average Precision (MAP)**, were employed. Precision focuses on the percentage of the number of truly relevant documents among those extracted, while recall makes this comparison based on all relevant documents within the system. Since there are more than 2000 unique documents in this case, precise calculation is impractical, leading to a manual approximation based on extracting and sampling the first twenty returned documents.

The **Average Precision (AvP)** is important because precision is what defines user satisfaction for the majority of users. In fact, users often do not require high recall since the percentage of relevant results given all important documents in the system is almost always unknown, unlike the relevance of the first returned documents. The precision evaluation focuses on the initial twenty documents (**Precision at K - P@K**) returned per query, as it reflects a balanced value aligned with typical usage patterns of a search engine.

The **Precision-Recall Curves** are constructed for each query and each system based on the subset of ranked documents returned. Ideally, a system is considered more stable the smoother its formed curve, and its performance is deemed better with a higher Precision-Recall Area Under the Curve [20]. This metric encapsulates the overall effectiveness of the system in balancing precision and recall across thresholds.

The **Mean Average Precision (MAP)** is a common metric used in Information Retrieval and represents the average of Average Precision metric across various sets returned over the evaluation period. It helps determine if the system is consistent even when applied to different information needs.

In the upcoming subsections, diverse user scenarios are presented as queries, accompanied by their respective results and statistics based on precision and recall metrics. The Table 18, provided in the annexes, documents the relevance of the top 20 results for each evaluated query and for the two systems under analysis.

A 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. The search is conducted using keywords like 'center London' within the review text, United Kingdom as location and the results are sorted in descending order based on their rating.

Query:

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

Table 5: Q1 information need results

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

Result Analysis: Both systems did well, although there is a notable increased precision on the boosted system, demonstrated in the table table 5. The utilization of eDismax, in this query, proved to be important for the results since the 2 words "center london", when put together, are correlated with each other.

According to Figure 14 and Figure 15, which represent the precision-recall curves of the simple and boosted systems, it can be observed that the boosted system exhibits superior performance. Indeed, it shows a larger area under the curve, and the decay of precision concerning the increase in recall is negligible in this context.

B Breakfast or Room Service

Information Need: Hotels with good breakfast or good room service in New Delhi.

Relevance Judgement: In this information need it is 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

Table 6: Q2 information need results

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

Result Analysis: The two systems have similar average precision, as it can be seen in table 6. 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 "Information Retrieval Improvements", section 8.

Both precision-recall curves of the systems, represented in figures 16 and 17, demonstrate a favorable trade-off between precision and recall. While there is a decline in the boosted system towards the end, it underscores the necessity for an alternative textual analysis.

C Convenience and Accessibility

Information Need: Hotel in the United Kingdom with good location and either elevator or good accessibility.

Relevance Judgement: With this query it is 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 country.

Query:

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

Table 7: Q3 information need results

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

Result Analysis: The systems differ in average performance, with the simple one overall performing better, as shown by the table 7. 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.

The curves precision-recall of the systems depicted in figures 20 and 21 indicate a degradation of the system concerning this query. On one hand, in the simple system, precision does not remain high as recall increases, progressively decreasing with a steep slope. On the other hand, the overall precision of the boosted system is quite low compared to other queries. This result once again emphasizes the need for a different type of textual analysis that considers the entire context of the sentence, to be explored in an upcoming section.

D 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

Table 8: Q4 information need results

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

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 B. 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.

figures 22 and 23 depict the precision-recall curves of the simple and boosted systems for this query, respectively. It is important to highlight that both curves exhibit a steep decline in precision as recall values increase, indicating a trade-off between these two metrics. The area under the curve is the smallest among the four conducted experiments, which is reasonable considering the complexity demanded by this query.

Taking into account all the results from the multiple information needs across queries, its presented in the following table the Mean Average Precision for both systems:

Table 9: Overall systems evaluation

Global	Syst. Simple	Syst. Complex
MAP	0.665	0.6975

Therefore, it is concluded that the system exhibits satisfactory performance, and, in general as anticipated, the more complex system is capable of yielding better results than the simple one.

8 INFORMATION RETRIEVAL IMPROVEMENTS

The previous stage introduced an initial version of the information retrieval system, evaluating it based on well-defined information needs and metrics grounded in precision and recall. Looking from another perspective, it also helped identify the weaknesses and limitations of the chosen approaches. Therefore, to enhance the search engine, this phase involved the implementation and evaluation of features aimed at addressing the identified gaps:

- Stopwords Filter
- Semantic Analysis

The evaluation of these features followed the same style as the previous ones, using real information needs as a base and adjusting them to the context encountered. There was still two systems in analysis, this time the boosted system and the boosted system with the application of the improvement under study. Evaluating each improvement separately allows for both an isolated analysis of its contribution to the system's success and a discussion on its permanence in the final search engine.

Three information needs were used. The first two were the same as in the previous stage, ensuring a valid comparison of the systems when exposed to the same environment and providing a visualization of the progressive development of the hypothesis. There was a comparison of the systems using a third information need, this time tailored to the improvement's objective, adding extra stress to the system on the topic we want to explore and checking the system's ability to handle the adversities of the natural language characteristic of these searches.

Tables 18 and 19 documents the relevance of the top 20 results for each evaluated query and for the two improvements.

In order to explore the project's theme more focusedly, More Like This [21] has been added to the list of improvements, with consequent analysis. Balancing the benefits of each of the topics addressed brings us even closer to what a contemporary information retrieval system looks like today.

8.1 Stopwords Filter

The decision to integrate a Stopwords Filter aimed at enhancing the exploration of larger phrases, allowing clients more flexibility to conduct searches with more complex queries. In the schema, the Stopwords Filter was applied both during the indexing process, affecting the internal storage of Solr, and in the query analyzer, where it played a role in eliminating irrelevant words during the search. Solr initially provides a set of predefined stop words, but a custom file was chosen, derived from a Python library [13]. Similar to the process for Synonyms, this custom stop words file was incorporated into the Solr configuration files.

Two queries from section 7 were reviewed in the following queries, each with a single stop word. When compared to the previous strategy, the new approach added two query parameters: stopwords and ignoreCase, both of which were set to true, indicating that stopwords would be investigated regardless of capitalization.

A Breakfast or Room Service

Information Need: Hotels with good breakfast or good room service in New Delhi.

Relevance Judgement: In this information need it is 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
- stopwords: true
- ignoreCase: true

Table 10: Q2 information need results

Rank	Previous Syst.	Improved Syst.
AvP	0.87	0.85
P@20	0.8	0.8

Result Analysis: Upon analyzing the results in the table 10, it is clear that the results of the query differ, as expected. Considering that the query only contains one stopword, the results show a noticeable similarity even though they are not identical. Upon examining the curves in figures 17 and 18, it is apparent that the area under the curve is lower than it was for the previous system. However, the latest ten findings show an increase in stability, indicating a minor decline in performance but more consistency in the results.

B 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
- stopwords: true
- ignoreCase: true

Table 11: Q4 information need results

Rank	Previous Syst.	Improved Syst.
AvP	0.55	0.59
P@20	0.5	0.55

Result Analysis: Very alike to previous analysis. Both systems exhibit similar average precision values, taking into account that the query only has one stopword.

Due to the complexity of the query, a more substantial improvement is evident compared to the previous one, as illustrated in figures 23 and 24. Both performance and stability have increased from the previous system to this one, highlighting its improvement.

C Beach Views

Information Need: Best hotels with beach views.

Relevance Judgement: In this task, the objective is to find the best hotels with beach views. Since we are using the Stopwords Filter, only the words "best hotels beach views" will be search in each review text.

Query:

- q: What are the best hotels with beach views
- q.op: AND
- fq: !child of="*:* -_nest_path_:*" location:*
- fl: *,[child]
- sort: score desc
- stopwords: true
- ignoreCase: true

Table 12: Q5 information need results

Rank	Previous Syst.	Improved Syst.
AvP	0	0.75
P@20	0	0.65

Result Analysis: In contrast, the third query, which has four stopwords, produced no results in the original schema without the stopwords filter. The modified schema, on the other hand, provided the expected outcomes with a precision of 0.75, as shown in the table 12.

The preceding model in this query exhibited zero performance, attributed to the number of stopwords present in the query. In contrast, the enhanced system demonstrated both stability and improved performance, as evidenced in figures 26 and 27 curves.

Observing the following table 13, the behaviour of the improved schema is better when increasing the complexity of the queries.

Table 13: Overall systems evaluation

Global	Previous Syst.	Improved Syst.
MAP	0.473	0.73

In conclusion, the integration of a stopwords filter provides the advantage of handling more complex queries, granting clients greater freedom in their search parameters. This enhancement contributes to the complexity and accuracy of the search results, ultimately improving the overall search system.

8.2 Semantic Search

The decision to implement Semantic Search emerged as a result of the evaluation of prior queries. Specifically, queries B and D revealed shortcomings, primarily attributable to the subjective nature of reviews. The challenges stemmed from instances where a single negative word could alter the overall context of the entire review, leading to multiple instances of failure.

To execute this integration, it was necessary to generate dense vectors that accurately captured the semantics of each word within the review text found in the JSON data populating Solr. This process also entailed adjusting our queries to smoothly align with the knn Query Parser [22], ensuring the retrieval of results closely associated with these generated vectors.

The evaluation incorporates the previously mentioned queries and introduced a new one focused explicitly on public transportation. Since there may be some reviews that reference specific public transports without relying solely on predefined keywords.

A Breakfast or Room Service

Information Need: Hotels with good breakfast or good room service in New Delhi.

Relevance Judgement: In this information need it is 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: !knn f=vector topK=20((good breakfast) OR (good room service) **vectorized**
- q.op: AND
- fq: !child of="*:* -_nest_path_:*" location: "new delhi"
- fl: *,[child]
- sort: score desc

Table 14: Q2 information need results

Rank	Previous Syst.	Improved Syst.
AvP	0.87	0.81
P@20	0.8	0.75

Result Analysis: Upon examining the results from the query in Table 14, it becomes evident that despite the similarity between the outcomes, the boosted system holds a slight advantage. Upon analyzing the reviews retrieved by the query, it is evident that contrary to expectations, semantic searches still encounter challenges when dealing with cases where the inclusion of a negative word alters the meaning of the sentence.

Although not immediately apparent, the area under the curve is in fact less than that of the previous system, as it's shown in figures 17 and 19. With only a slight drop in performance, the stability of the query is preserved due to its simplicity.

B 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: !knn f=vector topK=20((good vegetarian) OR (good vegan) **vectorized**
- q.op: AND
- fq: !child of="*:* -_nest_path_:*" location:*
- fl: *,[child]
- sort: score desc

Table 15: Q4 information need results

Rank	Previous Syst.	Improved Syst.
AvP	0.55	0.21
P@20	0.5	0.4

Result Analysis: Table 15 exhibits a behavior similar to the previous one, but with more pronounced distinctions. The semantic search demonstrates an even more significant under performance compared to the boosted system. Upon closely analyzing the reviews retrieved in this query, it becomes apparent that the positive context of the review amplifies the inclusion of cases where a word with negative connotations, such as "lack", alters the portion specific to our query.

The curves precision-recall of the systems depicted in figures 23 and 25 indicate a degradation of the system concerning this query. A constant value of 0.4 suggests a moderate level of precision throughout the range of recall values, indicating that a portion of relevant instances is being correctly identified but implying that a substantial number of irrelevant instances are being included in the results.

C Public transports

Information Need: Hotels with good access to public transports

Relevance Judgement: In this task, our goal is to identify hotels with convenient access to public transportation. Therefore, reviews should highlight proximity to nearby stations or accessible transportation routes.

Query:

- q: !knn f=vector topK=20(good access to public transports) **vectorized**
- q.op: AND
- fq: !child of="*:* -_nest_path_:*" location:*
- fl: *,[child]
- sort: score desc

Table 16: Q6 information need results

Rank	Previous Syst.	Improved Syst.
AvP	0.97	0.83
P@20	0.95	0.75

Result Analysis: Finally, Table 16 exhibits a pattern similar to its predecessors, where the boosted system consistently outperforms the semantic search.

Upon analyzing the retrieved results, it becomes apparent that, in this instance, the challenge does not stem from negations in the phrases. Instead, the semantic search tends to fetch reviews expressing the notion that there is no requirement for public transport as the hotel is conveniently situated in the midst of everything. Although this information may be of interest to users, it doesn't precisely align with the query.

As illustrated in Figures 28 and 29, the precision-recall curve for the semantic model exhibits lower performance compared to the boosted model. Additionally, there is a discernible lack of stability across the first 20 results indicating a less consistent performance.

Table 17: Overall systems evaluation

Global	Previous Syst.	Improved Syst.
MAP	0.80	0.62

In summary, Semantic Search falls short in effectively addressing the challenge posed by negative words altering the meaning of a phrase. This limitation, coupled with the complexities involved in its implementation, such as the need for query parsing, makes it not worth incorporating into the final product.

8.3 More Like This

Within Solr, the More Like This (**MLT**) functionality empowers users to discover documents similar to a specified document. Solr's More Like This is a Lucene [23] built-in functionality that operates by scrutinizing the text within the provided document and subsequently identifying other documents in the index that exhibit textual and contextual similarities. The outcomes of this query comprise documents with the most elevated similarity scores.

An approach to this feature is particularly suitable and crucial in the context of the current project. Given the inherent subjectivity in choosing hotels due to tourism, users are expected to want to find more areas or hotels aligned with their preferences.

Thus, from a hotel review, the system finds another 10 with similar content and corresponding hotels. These results are computed and ordered by Solr's internal match score. The parameters used in the queries were as follows:

- **mlt.fl = text**, the similarity comparison will always be based on the text between reviews, as this text contains the context of the evaluation;
- **mlt.mintf = 2**, (Lucene's default number), the minimum number of matches between terms for the document to be considered valid;
- **mlt.mindf = 5**, (Lucene's default number), the minimum number of valid documents for the search to be considered valid;

Although the More Like This feature allows for boosts in certain parameters, this was not utilized. It would not make sense to create an additional boost for the fields of the document being processed since we only applied the functionalities of MLT to the text field in the review. Applying the boost to specific terms or tokens would distort the concept of a global search system capable of analyzing each query in a balanced manner by favoring the emergence of some words.

The evaluation of this improvement does not follow the approach of previous information needs. The comparison of semantics and similarity between two texts can be even more complex and subjective when done manually. As such, we chose to evaluate the relevance of these queries using the objectivity of the Python SpaCy library [24]. SpaCy specializes in advanced natural language processing (**NLP**) and allows for the comparison and quantification of similarity between two texts through their context.

After collecting a sample from the system, where 20 reviews were randomly selected, along with the corresponding 10 reviews most similar according to Solr's output, the degree of similarity of their texts was externally computed. The result of the average similarity between the first ten results of each MLT query is described in Figure 9.

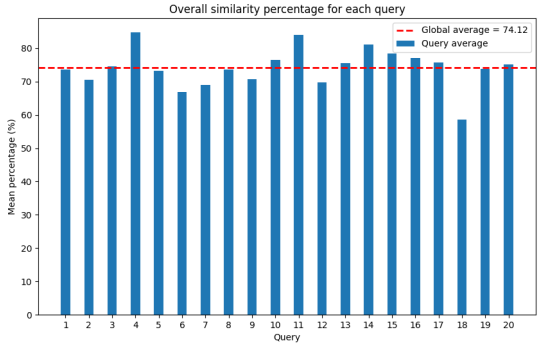

Figure 9: Average similarity percentage for each query

The overall average similarity between the queries is quite high, never falling below 60% in this sample. On the other hand, since it is a document ranking system, it is important to know the level of similarity decay with the index of the outputs. Theoretically, it is expected that the documents at the top have a higher degree of similarity than the others. To assess this behavior, the corresponding degree of similarity was computed for each index of the output. The mean similarity evolution across indexes can be found in Figure 10.

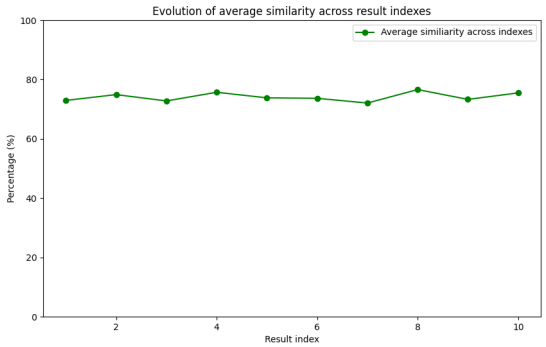

Figure 10: Evolution of average similarity across result indexes

Contrary to expectations, there was no inverse relationship in the selected sample between the degree of similarity of the original query and the index of the document. This fact can be justified in two ways. On the one hand, since the number of reviews in the dataset is much higher than the output of the MLT system, there was no spacing to notice a noticeable difference in the ranking of the results. On the other hand, the score from SpaCy is different from Solr's internal score, to which we do not have access, leading to some fluctuations.

With highly satisfactory results, this feature will be present in the final system.

9 SEARCH USER INTERFACE

While Solr's interface provides a valid approach to the search system and query customization, it deviates from traditional information retrieval system platforms. In this phase, the focus shifts to user interfaces by developing a frontend in NodeJS [25] using Solr's API [26] for the search system. This engine will empower travelers to explore and filter accommodations based on preferences, such as location, room quality, staff service, or other factors identified during the analysis phase.

Thus, four pages have been developed to support these needs, featuring a minimalist and responsive design for an enhanced user experience:

- **Home**, depicted in Figure 30, where users can initiate searches and preview the best hotels in the system;
- **Search**, illustrated in Figure 31, showcasing the results of the searches;
- **Hotel**, presented in Figure 32, displaying comprehensive information about the hotel, along with all its reviews;
- **More Like This**, shown in Figure 33, presenting results from similar searches;

It is also possible to apply filters to searches, in addition to specifying the query. Users can filter results by location, review rating given, and the hotel's average rate.

Each result is accompanied by all relevant information for assessing its relevance, such as the hotel's name, ratings, review text, and application date. All results are returned in descending order of ranking, ensuring that the first ones are the most relevant given the underlying context of the query and applied filters.

In order to assess overall qualitative aspects of the platform, a few students from the course were selected, and they were invited at the end of the experimentation session to respond to a brief questionnaire [27]. This questionnaire was developed using the standard version of the System Usability Scale - SUS [28]. This scale includes questions that evaluate the user's perception of the usability and overall relevance of the application, as well as the clarity of the implemented features:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Each response on the SUS scale is assessed on a scale from 0 to 5, resulting from the convergence of the subjective characterization of each topic from "Strongly Disagree" to "Strongly Agree". The results collected from the 11 students who volunteered for the interface evaluation can be found in table 21.

The collected results demonstrate that the users' evaluation of the interface according to the SUS scale averages 81.6%. Reference data for this type of analysis set 68% as the threshold for interface quality, so it is concluded that the implemented features meet the overall needs of the users. These positive results were driven by two distinct factors: on one hand, the nature of the selected users, who are students from the same course and therefore within the context of the needs of an application of this caliber, and on the other hand, the similarity and familiarity of the interface with other well-known research systems.

10 FINAL SYSTEM CHARACTERIZATION

After two iterations and their corresponding evaluations, the final information retrieval system has the following properties:

- **ASCII Folding** handles special characters and accents, converting them to their equivalent ASCII form
- **LowerCase** converts all characters to their lowercase counterparts.
- **Synonyms** expands each token to include variations based on its synonyms.
- **Stopwords** allowing more flexibility to conduct searches with more complex queries.
- **More Like This** allowing discover documents similar to a specified document.

In total, it ensures a global Mean Average Precision of the system of 73%. Additionally, it has a search user interface evaluated at 81.6% based on received feedback and the System Usability Scale.

11 CONCLUSIONS AND FUTURE WORK

In conclusion of this project, all planned tasks within the Data Preparation, Information Retrieval, and Improvement phases have been successfully completed. This achievement represents a pivotal moment in the process of creating a valuable hotel search engine that assists tourists in making well-informed choices.

One of the most challenging aspects of the work involved devising effective strategies for handling nested documents, along with their indexing and retrieval. Solr lacks documentation and concrete examples supporting the addressed document format.

However, there are always opportunities for improvement. Investing in the development of context-aware searches using broader NLP libraries, in addition to the features already implemented by Solr at this level, would be a winning strategy for a more globally applicable system with reduced bias.

With a final precision of 73% in terms of the retrieval system, and the capability to address various information needs within the chosen context verified, the developed architecture is the result of the theoretical application of knowledge acquired throughout the academic unit.

REFERENCES

[1] Apache solr, 2023. https://solr.apache.org/guide/6_6/introduction-to-solr-indexing.html, Last accessed on October 23, 2023.
[2] Kaggle, 2022. https://www.kaggle.com, Last accessed on September 30, 2023.
[3] Datafiniti, 2017. https://www.datafiniti.com, Last accessed on September 30, 2023.
[4] Datafiniti, 2017. http://www.ctan.org/pkg/acmart, Last accessed on September 30, 2023.

[5] Hotel reviews, 2017. https://www.kaggle.com/datasets/juhibhojani/hotel-reviews, Last accessed on September 30, 2023.
[6] Booking, 1996. https://www.booking.com, Last accessed on September 27, 2023.
[7] Reviews of london-based hotels, 2018. https://www.kaggle.com/datasets/PromptCloudHQ/reviews-of-londonbased-hotels, Last accessed on September 30, 2023.
[8] Datastock, 2018. https://datastock.shop, Last accessed on September 30, 2023.
[9] 515k hotel reviews data in europe, 2017. https://www.kaggle.com/datasets/jiashenliu/515k-hotel-reviews-data-in-europe, Last accessed on September 30, 2023.
[10] Pandas, 2023. https://pandas.pydata.org, Last accessed on October 9, 2023.
[11] Matplotlib, 2023. https://matplotlib.org, Last accessed on September 2, 2023.
[12] Numpy, 2023. https://numpy.org, Last accessed on September 2, 2023.
[13] Nltk, 2023. https://www.nltk.org, Last accessed on September 2, 2023.
[14] Information retrieval, 2023. https://link.springer.com/book/10.1007/978-3-319-93935-3, Last accessed on October 23, 2023.
[15] Solr tokenizers, 2023. https://solr.apache.org/guide/solr/latest/indexing-guide/tokenizers.html, Last accessed on November 2, 2023.
[16] Solr filters, 2023. https://solr.apache.org/guide/solr/latest/indexing-guide/filters.html, Last accessed on November 2, 2023.
[17] Indexing nested documents, 2023. https://solr.apache.org/guide/solr/latest/indexing-guide/indexing-nested-documents.html#schema-configuration, Last accessed on November 11, 2023.
[18] Queries in nested documents, 2023. https://solr.apache.org/guide/solr/latest/query-guide/searching-nested-documents.html, Last accessed on November 11, 2023.
[19] edismax, 2023. https://solr.apache.org/guide/7_7/the-extended-dismax-query-parser.html, Last accessed on November 4, 2023.
[20] Precision-recall area under the curve, 2023. https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html, Last accessed on November 7, 2023.
[21] More like this, 2023. https://solr.apache.org/guide/8_8/morelikethis.html, LastaccessedonDecember4,2023.
[22] Dense vector search. https://solr.apache.org/guide/solr/latest/query-guide/dense-vector-search.html,LastaccessedonDecember10,2023.
[23] Lucene, 2023. https://lucene.apache.org/core/9_9_0/index.html, Last accessed on December 8, 2023.
[24] Python spacy. https://spacy.io/, Last accessed on December 7, 2023.
[25] Nodejs. https://nodejs.org/en/about, Last accessed on December 10, 2023.
[26] Solr's api. https://solr.apache.org/guide/8_5/client-apis.html, Last accessed on December 9, 2023.
[27] Sus evaluation form. https://docs.google.com/forms/d/e/1FAIpQLSeaWvSApBxHiT_iXDgFF1eJ9HGKVdpSvcWV558pGX2PYa6Iqw/viewform,LastaccessedonDecember8,2023.
[28] System usability scale. https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html,LastaccessedonDecember8,2023.
[29] Solr stop filter, 2023. https://solr.apache.org/guide/solr/latest/indexing-guide/filters.html#managed-stop-filter, Last accessed on November 7, 2023.
[30] Natural language tool kit. https://www.nltk.org/,LastaccessedonDecember4,2023.

A ANNEXES

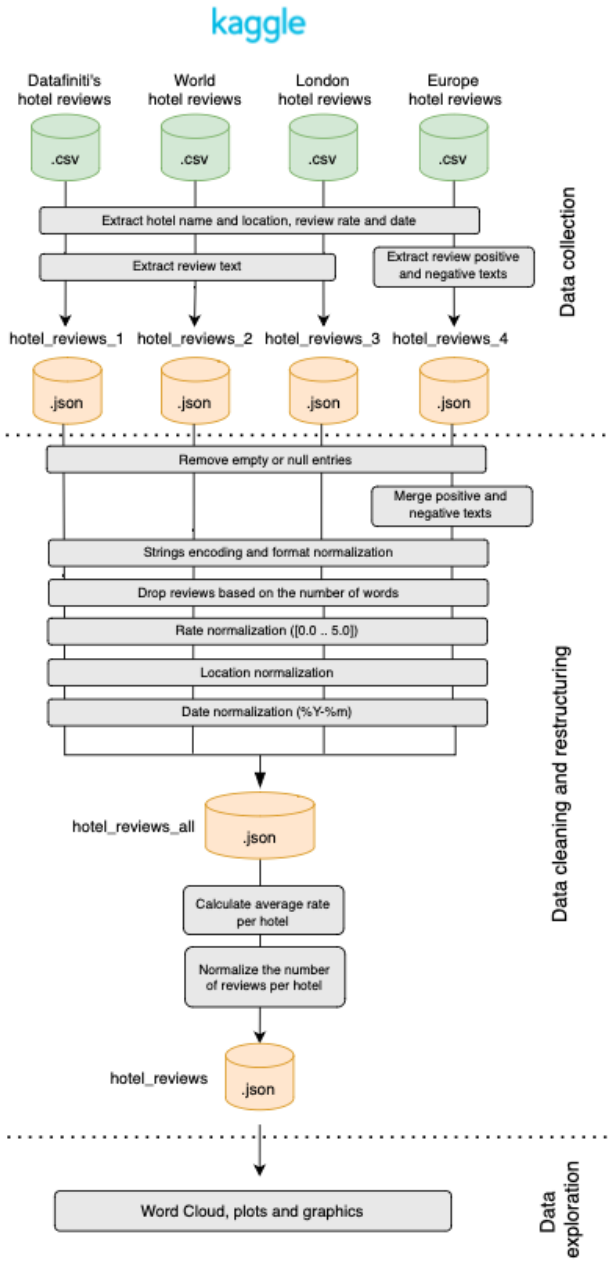

Figure 11: Data preparation pipeline

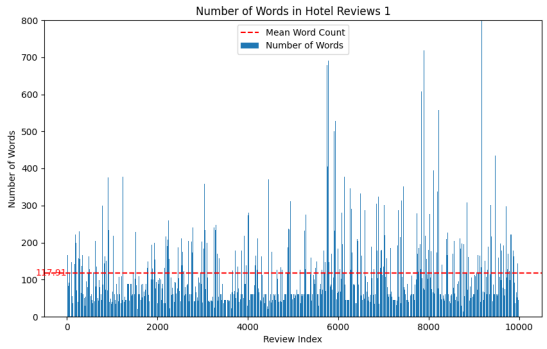

Figure 12: Average words per review in Datafiniti's Hotel Reviews dataset

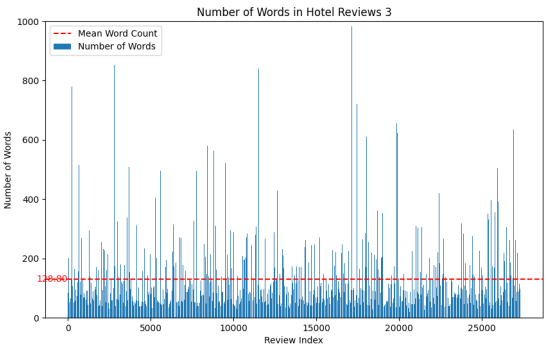

Figure 13: Average words per review in London Hotel Reviews dataset

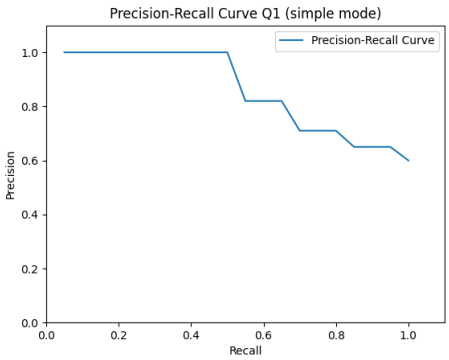

Figure 14: Q1 Precision-recall curve using simple system

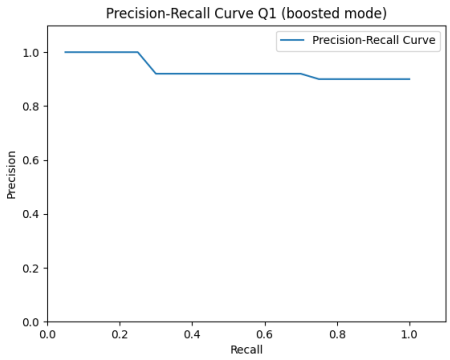

Figure 15: Q1 Precision-recall curve using boosted system

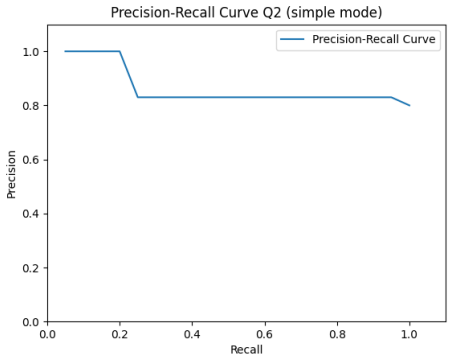

Figure 16: Q2 Precision-recall curve using simple system

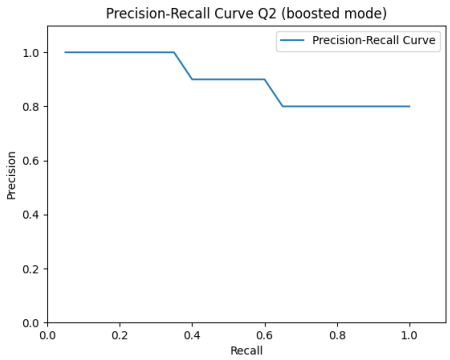

Figure 17: Q2 Precision-recall curve using boosted system

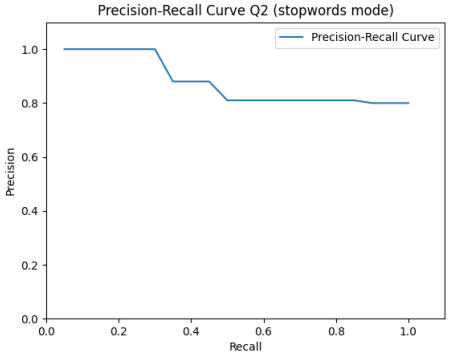

Figure 18: Q2 Precision-recall curve using stopwords filter

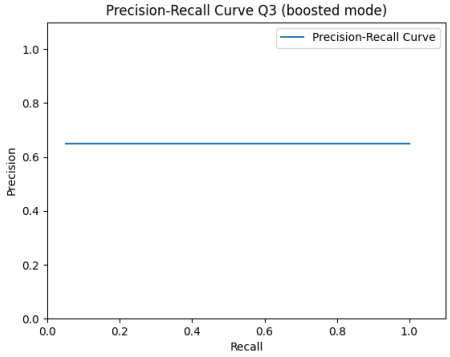

Figure 21: Q3 Precision-recall curve using boosted system

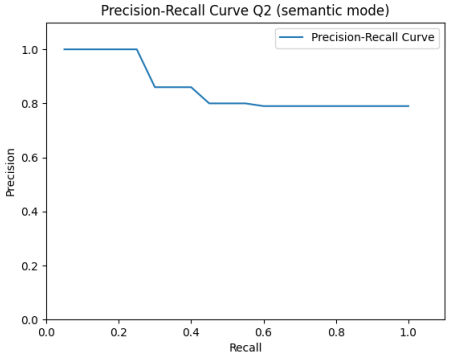

Figure 19: Q2 Precision-recall curve using semantic search

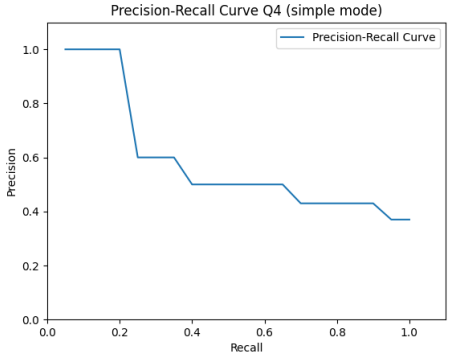

Figure 22: Q4 Precision-recall curve using simple system

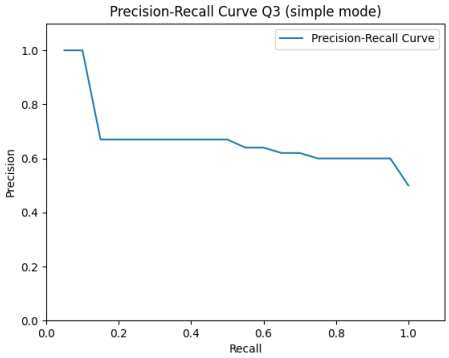

Figure 20: Q3 Precision-recall curve using simple system

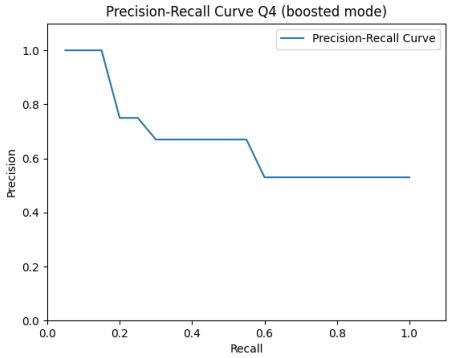

Figure 23: Q4 Precision-recall curve using boosted system

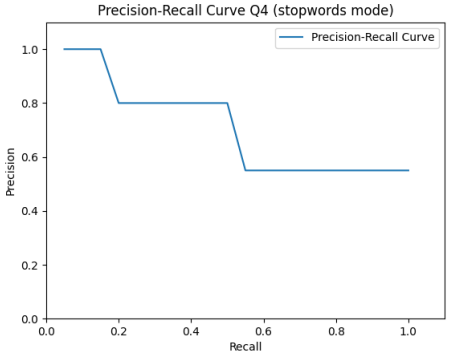

Figure 24: Q4 Precision-recall curve using stopwords filter

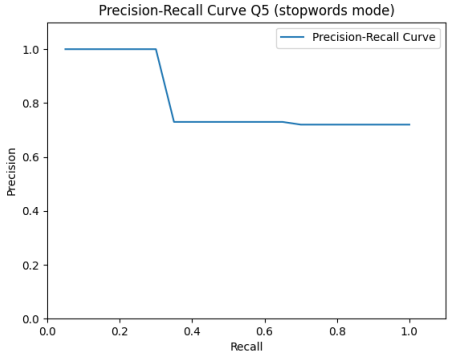

Figure 27: Q5 Precision-recall curve using stopwords filter

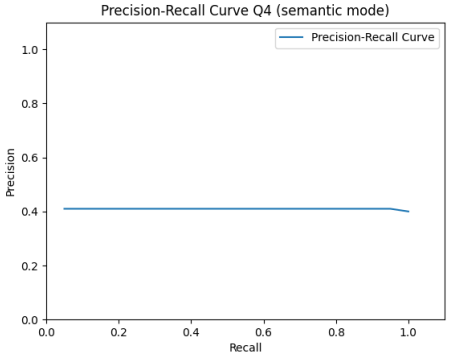

Figure 25: Q4 Precision-recall curve using semantic search

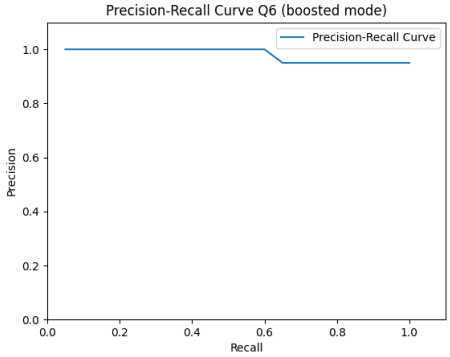

Figure 28: Q6 Precision-recall curve using boosted system

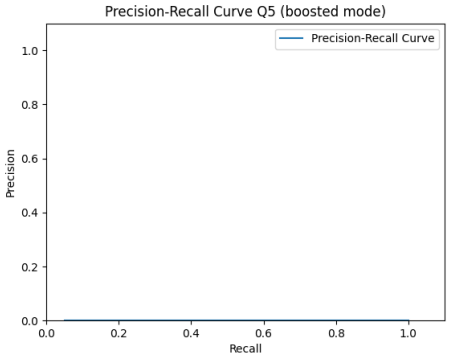

Figure 26: Q5 Precision-recall curve using boosted system

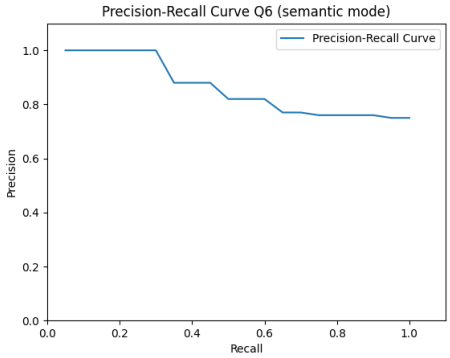

Figure 29: Q6 Precision-recall curve using semantic search

Table 18: Relevance results for initial system evaluation

Rank	Relevance Q1		Relevance Q2		Relevance Q3		Relevance Q4	
	Simple	Boosted	Simple	Boosted	Simple	Boosted	Simple	Boosted
1	1	1	1	1	1	0	1	1
2	1	1	1	1	0	1	1	1
3	1	1	0	1	1	0	0	0
4	1	1	0	1	0	1	0	1
5	1	0	1	1	0	0	1	0
6	1	1	1	1	1	0	0	1
7	1	1	1	0	1	1	0	0
8	1	1	1	1	1	0	1	0
9	0	1	0	1	1	0	0	0
10	0	1	1	1	0	1	1	0
11	1	1	1	0	1	1	0	0
12	0	1	1	0	0	1	0	1
13	0	1	1	1	1	1	0	0
14	1	0	1	1	0	1	1	0
15	0	1	1	1	1	1	0	1
16	0	1	1	0	0	1	0	1
17	1	1	1	1	0	0	0	1
18	0	1	1	1	0	1	0	1
19	0	1	0	1	0	1	1	1
20	1	1	1	1	1	1	0	0

Table 19: Relevance results for Stopwords Evaluation

Rank	Relevance Q2		Relevance Q4		Relevance Q5	
	Boosted	Stopwords	Boosted	Stopwords	Boosted	Stopwords
1	1	1	1	1	0	1
2	1	1	1	1	0	1
3	1	1	0	0	0	1
4	1	1	1	1	0	1
5	1	1	0	1	0	0
6	1	0	1	0	0	0
7	0	1	0	0	0	1
8	1	1	0	0	0	0
9	1	0	0	0	0	1
10	1	1	0	0	0	1
11	0	0	0	1	0	1
12	0	1	1	1	0	0
13	1	1	0	0	0	0
14	1	1	0	1	0	1
15	1	1	1	1	0	1
16	0	1	1	0	0	1
17	1	0	1	1	0	1
18	1	1	1	0	0	1
19	1	1	1	1	0	0
20	1	1	0	1	0	0

Table 20: Relevance results for Semantic Search Evaluation

Rank	Relevance Q2		Relevance Q4		Relevance Q6	
	Boosted	Semantic	Boosted	Semantic	Boosted	Semantic
1	1	1	1	0	1	1
2	1	1	1	0	1	1
3	1	1	0	0	1	1
4	1	1	1	0	1	1
5	1	0	0	0	1	1
6	1	1	1	0	1	0
7	0	1	0	1	1	1
8	1	0	0	0	1	1
9	1	1	0	1	1	0
10	1	1	0	0	1	1
11	0	0	0	1	1	1
12	0	1	1	0	0	0
13	1	1	0	1	1	1
14	1	1	0	1	1	0
15	1	1	1	1	1	1
16	0	1	1	0	1	1
17	1	1	1	1	1	1
18	1	1	1	0	1	0
19	1	1	1	0	1	1
20	1	0	0	1	1	1

Table 21: SUS results

User	1	2	3	4	5	6	7	8	9	10	Points
1	3	2	5	3	4	1	4	2	4	2	75
2	4	1	5	3	4	2	4	1	4	2	80
3	3	2	5	2	4	2	5	1	4	2	80
4	2	1	5	2	5	2	5	2	4	1	82.5
5	5	1	4	2	4	2	5	1	3	1	85
6	5	2	4	2	5	3	3	2	4	2	75
7	3	2	5	1	4	1	4	2	5	3	80
8	5	2	4	1	3	2	4	1	5	1	85
9	4	1	5	2	4	1	5	1	4	2	87.5
10	3	2	4	1	5	2	5	2	4	2	80
11	4	1	4	1	5	2	4	1	5	2	87.5

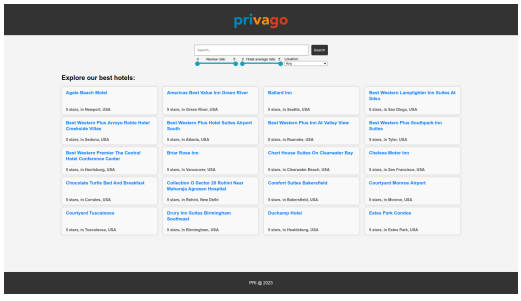

Figure 30: Homepage Interface

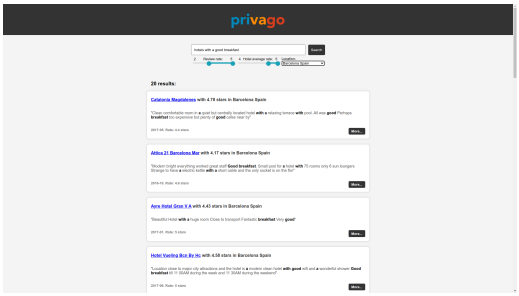

Figure 31: Search Page Interface

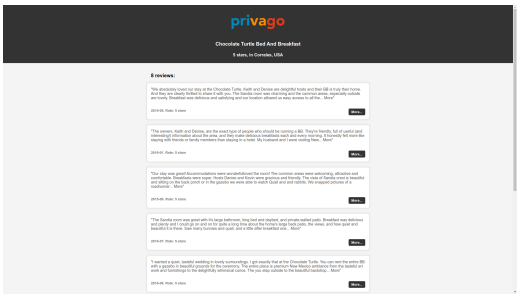

Figure 32: Hotel Page Interface

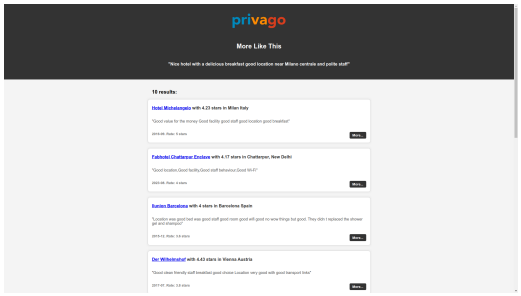

Figure 33: More Like This Page Interface