Pitchfork Music Reviews

Fabio Huang
Faculty of Engineering of the University of Porto
Porto, Portugal
up201806829@up.pt

Lisa Sonck
Faculty of Engineering of the University of Porto
Porto, Portugal
up202202272@up.pt

Abstract

Information communication technologies are ubiquitous in modern societies. An ever-growing number of activities depend on the ability to extract value from information. With this in mind a information retrieval system is built over a dataset of music reviews. The developing happened in three phases, the Milestones. The goal of the first milestone was to prepare, explore and characterize a selected dataset. As a result, a reproducible pipeline of data processing is achieved. In our case, a dataset about music reviews on Pitchfork (site best known for its daily output of music reviews) was retrieved from Kaggle. The data is analysed and explored using OpenRefine, which was then cleaned and refined using Python's Pandas Library [2]. To characterize the data, graphs about the data were formed and analysis was conducted. The second milestone was about Information Retrieval. For this the dataset was indexed using filters and tokens and represented in the tool Solr. To answer the defined information needs in milestone 1, queries were developed and executed on the system during milestone 2. These results were evaluated to see how good the indexed system can retrieve specific data. This already gave some accurate results, but improvements could be made. These improvements were executed during milestone 3. Here 4 extra information needs were, so now a total of 6 needs are presented. Besides this 6 systems were developed, each time an extra kind of improvement was added. the improvements include extracting keywords and entities, the use of synonyms, the polarity of the content and changing the content of the review, by deleting entities, titles and artist names. These different systems are tested together with the information needs and the results were analysed. From this we see that the system that uses the review_content where the entities, the title and the artist name is removed, with keywords extracted and uses the extended schema has the best mean average prediction of 80%. The use of synonyms still had some trouble with relevance, but by improving this and adding different boost, the whole system could be improved more.

Keywords: Information Processing and Retrieval, Datasets, Pitchfork Music Reviews

1 Introduction

The amount of available data is growing every year [1] and with it, the importance of its correct management. Being able to efficiently use the data and extract interesting results is essential for further development of our current society. The course Information Processing and Retrieval at FEUP aims to enlighten us about information search systems and their use for large amounts of data. To put this knowledge into practice, we were tasked to develop a search system, with the corresponding tasks of data collection and preparation, and the information processing and retrieval. This project is divided into three milestones: 1. Data Preparation, 2. Information Retrieval and 3. Building the final Search System. The first part of the report elaborates on the results and actions of the first milestone, which include:

- Choosing a dataset;
- Assessing the data quality and the authority of the source;
- Building a data processing pipeline;
- Cleaning of the data;
- Making a conceptual model of the data;
- Performing an exploratory data analysis;
- Characterizing the data;

The second part of the report talks more about the system built in Milestone 2 and the executed queries. The specific topics are:

- Solr;
- Our used Documents;
- The indexing procedure, with the used filters and tokens;
- The different evaluated queries with their results;

The third part of the report talks about the systems we built in Milestone 3, and the conclusions we arrived throughout this project. The topics we will discuss in this section are:

- The Improvements;
- Used Systems;
- The Information Needs, analysing and evaluation their results in the systems;
- Final Conclusion;

2 M1 - Data Preparation

2.1 Data Collection

Developing a information search system requires a good and extensive dataset. The dataset should contain a lot of rows and columns, where at least one has an exhaustive, diverse body. This way text search can be implemented and interesting results can be obtained.

2.1.1 Dataset Choice.

The choice of a dataset is important, since it determines the course of the project. At first the theme 'Music' was chosen, since a lot of large and comprehensive datasets exists for this subject and we have a big interest in music and its characteristics. We first found datasets were about artists, albums and songs, and the lyrics associated with each song, however, due to copyright reasons we were not able to get the complete content of the lyrics, and faced the problem of lacking textual data to work with. Luckily we managed to find a dataset that contained music reviews, which consisted of interesting comments and rich text that could be used for this project. [4]

2.1.2 Dataset Content.

The selected dataset consists of a .sqlite file that contained the following tables:

- Artists table: contained the name of the artists (18.8k rows)
- Content table: contained the review's content (18.4 rows)
- Genres table: contained the song's genre (22.7k rows)
- Labels table: contained the album's label (20.2k rows)
- Reviews table: contained the title of the review, name of the artist, url link for the pitchfork's review page, rating score of the review, best new music, review's author name, author type, review's publication date and week day (18.4k rows)
- Years table: contained the year that the song was released (19.1k rows)

2.1.3 Data Quality and Source.

Our dataset was retrieved from Kaggle [4], which is a popular community for data researchers and analysts and where the vast majority of the content is reliable. All of the content of this dataset was scraped from Pitchfork and since this dataset has over 10k downloads and over 8000 upvotes, we can infer that this source is reliable. The quality of the data meets our usage standard.

2.2 Pipeline

To build our Pipeline, we created python scripts using libraries such as Pandas [2], Matplotlib [5] and Seaborn [6] to manipulate the data and to generate graphs. OpenRefine [7] was also used to check for data facets. Our pipeline begins with a .sqlite file extracted from Kaggle, which contains multiple tables, as mentioned before. These tables are merged and a .json file is produced. This file is then processed and cleaned. At the end of the pipeline, the final dataset is used for data characterisation.

2.3 Data Cleaning

Before cleaning the data, we used OpenRefine [7] to explore the text facets on each column. After having an overall idea of the values of each column, duplicated and irrelevant columns were removed. Also entries with an empty review content were removed. At last some columns were renamed for an easier interpretation of the values.

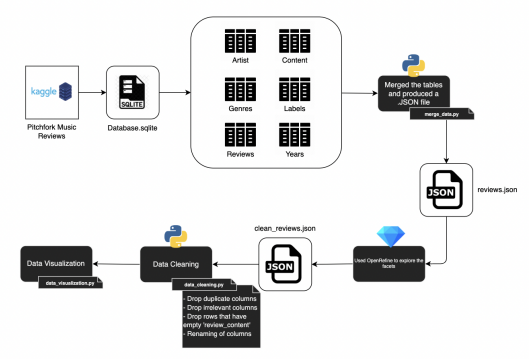


Figure 1. Pipeline Diagram

2.4 Conceptual Model

The conceptual model is based on the initial database, extracted from Kaggle. However, some of the attributes are removed after the data was refined and cleaned. The final model can be seen in figure 2.

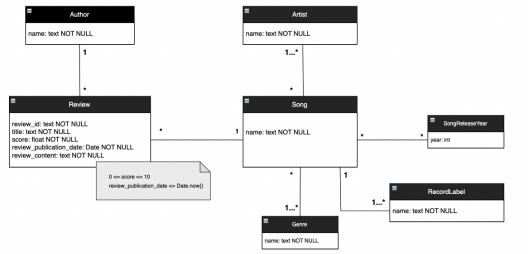


Figure 2. Conceptual Model

2.5 Dataset Characterisation

To better understand and grasp the data at hands, analysis and characterisation of the dataset was performed. To execute and visualize the characteristics of the dataset, the language Python was used with the packages Pandas [2] and Seaborn [6]. Pandas is very useful to handle big amounts of structured data and with Seaborn, graphs can be made easily. At first all the fields in the data set were investigated in a general way; Look at the possible values, what is the average value, are their outliers, etc. This initial research showed that the average score, given in a review, is 7.0 and the median is 7.2.

In figure 3 the number of reviews per score are represented. For every review the score is rounded to the nearest integer and this shows that the score eight is given the most, followed by seven. After investigation of the genre of the songs of

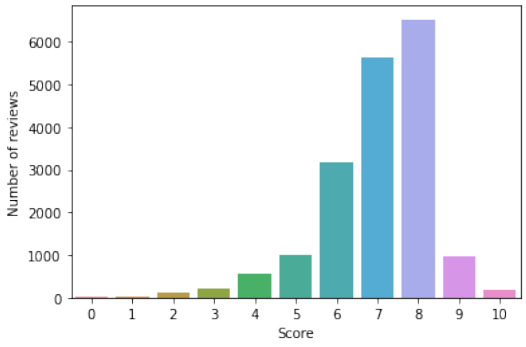


Figure 3. Number of reviews per rounded score.

the different reviews, we concluded that most of the reviews are written of rock songs. This is seen in figure 4. This first

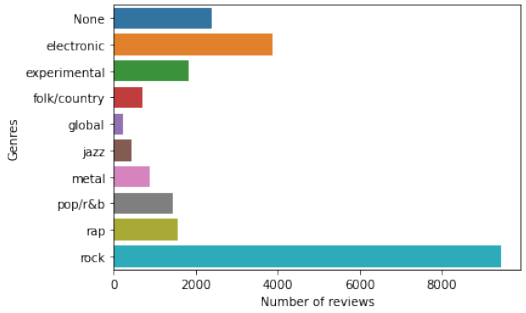


Figure 4. Number of reviews per song-genre.

analysis about the data gave a general overview about its characteristics and how our data set was built. To see if certain fields of the data had a correlation, cross analysis over the different fields was performed. Figure 5 shows that the average score is approximately the same for every genre and equal to 7.0, which also establishes the analysis of the score over all the reviews.

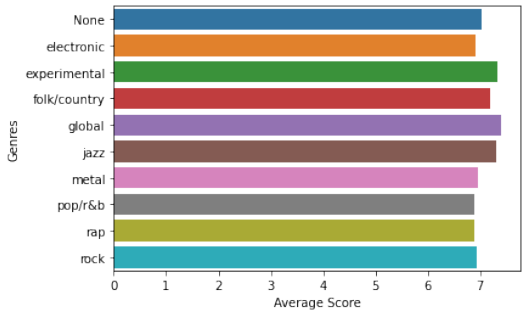


Figure 5. Average score per genre.

To have an overview about the length of a review, an analysis about the amount of words within one review was executed. In figure 6 the density of the amount of words in a review is plotted. This shows that most of the reviews consist of approximately 700 words. This is also concluded from the average word count and median, which are respectively equal to 703 and 649. To be able to make a representative further analysis, all the reviews who have a word count higher than 1500 are considered together

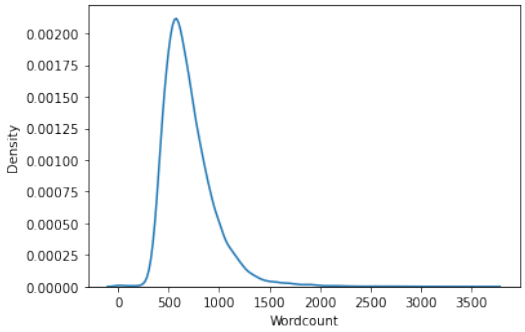


Figure 6. Density of the amount of words in a review.

As seen in figure 7 reviews for every genre have approximately the same amount of words. However, figure 8 shows an interesting result. It shows that the higher the word count of a review, the higher the average score is, with a difference of two points, on a scale of 10, between the shortest and longest reviews.

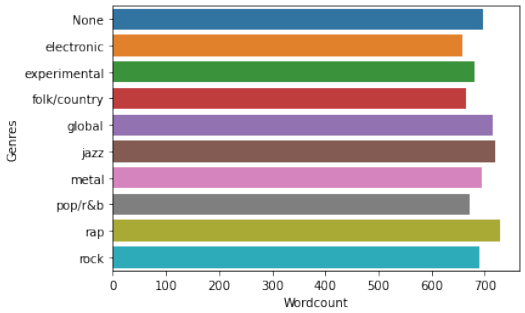


Figure 7. Average amount of words of a review per genre.

Besides this, text analysis of the reviews has been executed, where the most common words of reviews with a score above the average are found. Also for reviews with a score above the average the most common words are searched for. These findings can be further explored during the next milestone

2.6 Search tasks

After retrieving, cleaning and characterizing the data, insights about the properties of the data were acquired. With these insights, it was possible to devise useful queries for our search system.

- I want to find reviews of albums that include jazz and piano.

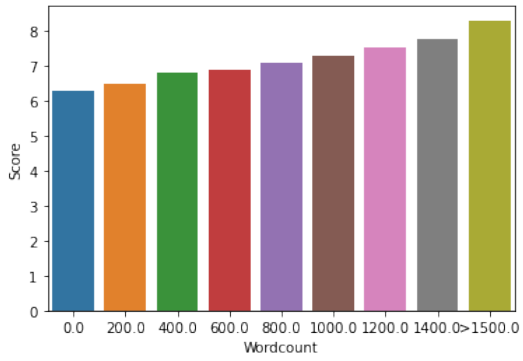


Figure 8. Average score of a review based on its word count.


```
{
  "reviewid":[13088],
  "title":["sketches of spain: legacy edition"],
  "artist":["miles davis"],
  "rating_score":[8.0],
  "author":["mark richardson"],
  "review_publication_date":[1244160000000],
  "genre":["jazz"],
  "song_release_year":"2009",
  "record_label":["legacy"],
  "id":"bd92093a-9f2b-4405-9a75-80ba845e12a4",
  "review_content":["Is this even jazz? Sketches of Spain was p
  "_version_":1749576559401893888,
  "score":8.093981},
{
```


Figure 9. Example of a document.

Type	Field	Indexed
standard_text	artist	TRUE
	author	TRUE
	genre	TRUE
	song_release_year	TRUE
	record_label	TRUE
pdoubles	rating_score	TRUE
plongs	review_publication_date	TRUE
free_text	review_content	TRUE
	title	TRUE
string	reviewid	FALSE

Figure 10. Schema Fields.

There were field which used the Solr default field type, in case, 'review_publication_date' and 'reviewid' that were defined using the types 'plongs' and 'string' respectively. Custom field types were also defined, specifically 'standard_text' and 'free_text', which both contained the tokenizer solr.ClassicTokenizerFactory, additionally, different filters were also applied to each of them, which can be visualized in the figure below. Each filter has a different functionality, which alters the way the tokens can be used in a query [9].

Field Type	Filter	Tokenizer
standard_text	solr.ASCIIFoldingFilterFactory	solr.ClassicTokenizerFactory
	solr.LowerCaseFilterFactory	
free_text	solr.ASCIIFoldingFilterFactory	
	solr.LowerCaseFilterFactory	
	solr.EnglishMinimalStemFilterFactor	
	solr.EnglishPossessiveFilterFactory	
	solr.HyphenatedWordsFilterFactory	
	solr.KStemFilterFactory	

Figure 11. Schema Field Types.

- Returns a list of reviews where the albums contains jazz and piano.
- I want to find albums that are reviewed as boring.
  - Returns a list of albums where the review_content contains the word boring
- I want to find songs and albums that are ideal for studying.
  - Returns a list of albums where the review_content contains words like calm, quiet, ambient, etc.
- I want to find albums that contain a drum solo.
  - Returns a list of albums where the review_content contains the words "drum" and "solo".
- I want to find reviews where Bob Dylan is mentioned but is not the artist.
  - Returns a list of albums where the review_content contains the words "Bob" and "Dylan" and where Bob Dylan is not the artist.
- I want to find reviews where the author praises the artist's voice
  - Returns a list of albums where the review_content contains the author praising the voice of the artist.

3 M2 - Information Retrieval

3.1 Used Documents

One review, together with all it's information, was used as a document in Solr. An example of a document, with all it's attributes, can be seen in figure 9. This choice of documents was made, because the main topic of our search system are the reviews. With the use of these documents all information regarding one review is kept together and querying over them is made easier.

3.2 Indexing

During the indexing process, we indexed most of our fields, except from the one that would not contribute for our information needs. Below in the figure 10, shows the schema that we have implemented:

3.3 Query 1: Boring songs

The first question answers the information need of finding albums that were criticized as boring by the authors of the reviews. The query searches for the word "boring" inside the review_content. If boring is found, the result will count as 5

times in the end-score, so reviews that often name the word "boring" are higher in rank. Besides this the query penalizes when the album has boring in the title, since this alters correct results and when boring occurs in the review_content near "never", "not" or "doesn't". The query also favors when "is boring" occurs in the review_content. At last the query also penalizes the results if boring is near a quotation mark, because this means that the word boring is not used as the exact definition of the word. Below the query insert in the q-field in Solr can be found.

```
review_content:boringˆ5
-title:boring -review_content:"doesn't boring" ~
5
-review_content:"not boring"~ 5
-review_content:"never boring"~ 5
review_content:"is boring"~ 5
-review_content:" "(")" boring "(")" "~3ˆ2
```

3.4 Obtained results.

This query was applied on our system, one time with the schema, as discussed in section 3.2, and one time without schema, where Solr decides the indexing of documents automatically. In table 1 the values of the calculated metrics for our system can be found. The average precision was calculated, together with the precision at 5, 10, 15, 20 and 15 fetched documents. The values were determined for both setups.

Metric	Values With Schema	Values Without Schema
Average Precision	0.719355	0.740282
Precision at 5 (P@05)	1.0	0.8
Precision at 10 (P@10)	0.6	0.7
Precision at 15 (P@15)	0.6	0.666667
Precision at 20 (P@20)	0.65	0.65
Precision at 25 (P@25)	0.68	0.56

Table 1. Results with and without an applied schema.

Figure 12 shows the precision recall curves for both setups. Precision tells how many of the retrieved documents were relevant and recall tells how many of the relevant documents were retrieved.

3.4.1 Evaluation.

When our schema was applied, the query returned 1114 results and without schema only 576 documents were retrieved. This is because of the usage of the different filters in our free-text filed-type. All words related to boring, e.g. bored, are also considered when executing the query. Because these amounts of documents are too much to evaluate manually, only the first 25 results in both setups are taken into account when calculating the metrics.

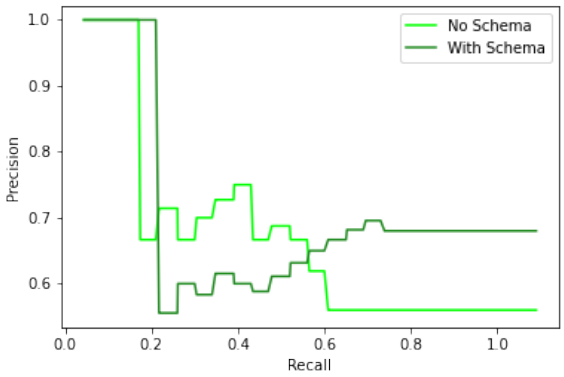


Figure 12. Precision recall curves of the results with and without an applied schema.

Using the schema yields a better precision for the first few results, as seen in table 9. Until 5 retrieved documents, the system with a schema has a precision while the system without a schema only has a precision of 80%. After that the precision of the system with schema drops suddenly to 60%. This is because the word bored or boring is used in sarcastic way, so not with it's original meaning. Our filters cannot handle these cases and thus yield a faulty result. The system without a schema also suffers from these cases, but only for the cases with boring and not bored. The moment an author uses a sarcastic comment with boring or bored, he often does it again, to illustrate how exciting the song is, which gives these results a high ranking in the query.
After these cases the precision of our system with schema increases again with documents that use the word boring in it's literal meaning. All kinds of words related to boring are taken into account in the system with schema, which yields a higher precision compared to the system without a schema. Without a schema only the word boring is taken into account and when it is used in another context than to describe the album, it has a higher influence on the results than in the one with schema, since no other words can be used as proof.
The precision-recall curves of both setups, as seen in figure 12 show a similar conclusion. The setup with schema reaches a recall of 0.2 with a precision of 1.0, while without a schema only a recall of less than 0.2 can be reached with precision 1.0. This means that the more first results are relevant in the system than in the system without schema. Between a recall of 0.2 and a little less than 0.6, the precision is better for the system without a schema. This can be explained by the use of sarcasm in the reviews (and that with schema the system is more sensitive to it, since it takes more words related to boring into account). When the recall reaches 0.6, the precision for the system with schema has again a better precision. It

reaches a precision of 68% when 25 documents are retrieved, compared to 56% for the setup without a schema. This means that more relevant documents were retrieved in total for the system with schema.

3.5 Query 2: Songs for studying

The second question answers the information need of songs and albums which are ideal for studying. Although it depends person to person which kind of music is ideal for studying, here, we consider them to be songs that are calm and peaceful. The query searches for the words "calm", "quiet", "ambient", "relaxing" and "classical music" that are present inside the review_content and exclude the word "violent". We favor "jazz" above the others, because jazz music tend to be more appropriate compare to the other genres available, which are (Rock, Electronic, Hip-Hop, R&B, Country...). After the query is done, we filter the ones that are considered to be really good (in terms of rating score).

q:

- genre:jazz^2
- review_content: calm
- review_content: quiet
- review_content: ambient
- review_content: relaxing
- review_content: "classical music"
- !review_content: "violent"

fq:

- rating_score: [8.0 TO 10.0]

3.5.1 Obtained results.

In table 18 the values of the calculated metrics for our system can be found. The average precision was calculated, together with the precision at 5, 10, 15, 20 and 15 fetched documents. The values were determined for both setups.

Metric	Values With Schema	Values Without Schema
Average Precision	0.475046	0.48338
Precision at 5 (P@05)	0.6	0.8
Precision at 10 (P@10)	0.6	0.6
Precision at 15 (P@15)	0.466667	0.466667
Precision at 20 (P@20)	0.35	0.35
Precision at 25 (P@25)	0.28	0.28

Table 2. Results with and without an applied schema.

Figure 13 shows the precision recall curves for both setups for query 2.

3.5.2 Evaluation.

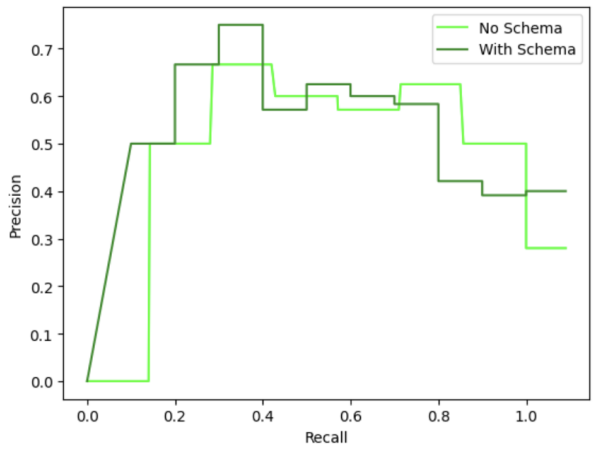


Figure 13. Precision recall curves of the results with and without an applied schema for query 2.

Since the results were quite similar and difficult to distinguish which one was better, the AUC (Area Under Curve) was calculated:

System	AUC value
Query 2 Without Schema	0.4964
Query 2 With Schema	0.5309

Table 3. AUC for Query 2.

From the table 3 above, we can conclude that the system with Schema does indeed has a slightly improved compared with the one without

3.6 Changes to Milestone 1

The search tasks in Milestone 1 were not optimal after the first iteration. We changed these during the second Milestone, so the information needs are more clear and querying could be done in a more optimized way.

4 M3 - Search System

During milestone 3 it was the goal to improve our system, so more relevant results could be obtained. The main implemented improvements are extracting keywords from the review_content, extracting the entities from the review_content and introducing synonyms for the keywords.

4.1 Improvements

4.1.1 Extracting the keywords.

The keywords are extracted from the review_content. To implement the extraction Python [10] was used as the programming language, together with the packages Pandas [2], spaCy [11] and Collections [12]. With spaCy the ”en_core_web_lg” pipeline is loaded, which is pretrained to separate a text

in tokens. In figure 14 this pipeline is named nlp. The review_content goes through this pipeline, which results in tokens. Only the tokens that are Adjectives, Nouns or Verbs are kept, with the exception of stopwords and words specific for the music industry. When all the relevant tokens are gathered, the Counter of collections is used, to find the most common keywords. Only the 25 most common are kept in in the dataset. The number 25 is chosen, since less keywords often result that words that occur frequently (+2 times) are not taken into account and with more words the extracted keywords would also contain words that only occur once. 25 is the best limit for the optimised use of keywords.

```
def get_hotwords(text, x, title, artist):
    result = []
    pos_tag = ['PROPN', 'ADJ', 'NOUN', 'VERB']
    doc = nlp(text.lower())
    for token in doc:
        if(token.text in nlp.Defaults.stop_words or token.text in punctuation):
            continue
        if(token.text in music_words):
            continue
        if(token.pos_ in pos_tag):
            result.append(token.text)
    out = Counter(result).most_common(x)
    return out
```

Figure 14. Function used to extract keywords from the review_content

4.1.2 Extracting the entities.

Entities are “real-world objects” that are assigned a name. These entities can also be extracted from the review_content by the use of Pandas [2] and spaCy [11]. Like with extracting the keywords, the extraction of entities also happens with the pretrained pipeline ”en_core_web_lg”. The entities are subcategorized in ’organizations’, ’places’, ’persons’, ’facilities’, ’law’, ’work_of_art’, ’events’, ’products’ and ’Nationalities or religious or political groups (norp)’. Per review these entities are extracted and divided according to their category. Figure 15 shows an example of such extracted entities.

organizations	places	persons	facilities	law	work_of_art	events	products	norp
[Pre-Millennium Tension, Massive Attack, Mezza...	[Bristol, Jamaica, Bristol, Memphis, UK, Earth...	[Portishead, Geoff Barrow, Portishead, Robert ...	[Del Naja]	[]	[The Conversation's Gene Hackman, Inertia Cree...	[]	[]	[]
[Krallice, Krallice, releases—2015, Hyperion, ...	[New York, Prelapsarian, Prelapsarian]	[Colin Marston, Mick Barr's, Lev Weinstein's, ...	[the Towers of Terror]	[]	[Transformation Chronicles, Lotus Throne, Hate...	[the Winter Solstice]	[]	[]
[Uranium Club, Human Exploration, Them Natural...	[Minneapolis, Detroit]	[Devo, teddy]	[]	[]	[Operation Pt, The Lottery, Opus, That Clown's...	[]	[]	[British]

Figure 15. Example of extracted entities of reviews.

4.1.3 Sentiment Analysis.

Sentiment Analysis enables us to quantify, extract and identify the effective states (positive, negative, or neutral), which will help us understand the sentiment of the author when writing the review. In order to do this, a python library called ’TextBlob’ [13] is used, which allows us to find not only the polarity of the text but also the subjectivity. However, the polarity results obtained are quite neutral, which can be visualised in figure 16. This can be explained by two main reasons, one is that Pitchfork reviews are generally neutral, and two is that our reviews contain a lot of text, which also makes analysis more difficult and results will tend to approach neutral. Unfortunately, the data obtained from this analysis was unusable, so it was not added to our dataset.

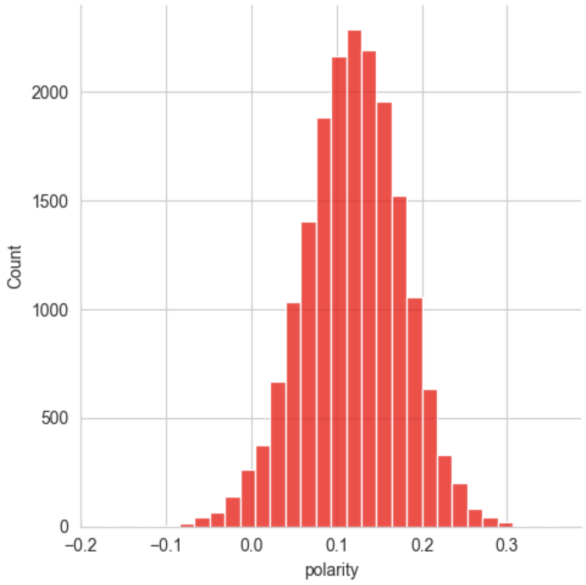


Figure 16. Polarity for reviews

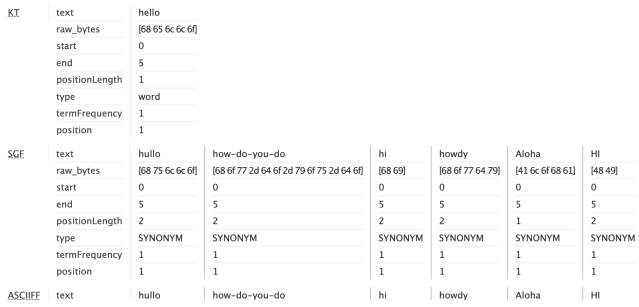
Note: Polarity values ranges between [-1.0, 1.0]. -1 being negative and 1 being positive.

4.1.4 Introducing synonyms.

After we extracted the keywords from the reviews, we introduced synonyms to these keywords. In order to do it, we need a text file that contains the list of synonyms that we want to use (on each line of this file we specify the words that are synonyms, separated by a space), in our case, we chose to use an existing database of synonyms from a Github Repository called Thesaurus [15]. After the synonym file made available, we applied the filter *’solr.SynonymGraphFilterFactory’* [14] to the keywords, this filter also enables us to specify the path of the file that stores the synonyms that we got using the parameter *’synonyms’*. After the setup was done, we should be able to search for keywords with synonyms 17, as in the figure below.

4.2 Used Systems

To be able to accurately evaluate the different improvements, six different systems were developed, as seen in table 4. System_1 uses a basic schema and the dataset with original



KT	text	hello					
	raw_bytes	[68 65 6c 6c 6f]					
	start	0					
	end	5					
	positionLength	1					
	type	word					
	termFrequency	1					
	position	1					
SGF	text	hullo	how-do-you-do	hi	howdy	Aloha	HI
	raw_bytes	[68 75 6c 6c 6f]	[68 6f 77 2d 64 6f 2d 79 6f 75 2d 64 6f]	[68 69]	[68 6f 77 64 79]	[41 6c 6f 68 61]	[48 49]
	start	0	0	0	0	0	0
	end	5	5	5	5	5	5
	positionLength	2	2	2	2	1	2
	type	SYNONYM	SYNONYM	SYNONYM	SYNONYM	SYNONYM	SYNONYM
	termFrequency	1	1	1	1	1	1
	position	1	1	1	1	1	1
ASCIIFF	text	hullo	how-do-you-do	hi	howdy	Aloha	HI

Figure 17. Keywords Synonyms Analysis using Solr example

content and no keywords. System_2 also uses the dataset with original content and no keywords, but uses the extended schema. Also System_3, system_4 and system_5 use the extended schema, but all have a different dataset, respectively the original content, the entity-free content and the clean content, each with associated keywords. System_6 also uses the clean content with keywords, but uses the synonym schema. All the different schemas and datasets are explained below.

	Basic Schema	Extended Schema	Synonyms Schema
Original Dataset without Keywords	system_1	system_2	
Original Dataset with Keywords		system_3	
Entity-free Dataset		system_4	
Cleaned Dataset		system_5	system_6

Table 4. Used systems.

4.2.1 Original Dataset. The Original Dataset uses the dataset of Milestone 2, with the fields artist, author, genre, song_release_year, record_label, score, review_publication_date, review_content, title and reviewid. The review_content is the same as can be found on the website of Pitchfork [26]. When the Original Dataset is used with keywords, it means that the keywords are extracted from the untouched review_content and that entities are extracted. If the Original Dataset is used without keywords, it means that no keywords or entities are retrieved from the review_content and those fields are just empty.

4.2.2 Entity-free Dataset. The Entity-free Dataset is made by changing the review_content of the Original Dataset. All the extracted entities of the original review_content are deleted from the review_content in the Original Dataset and from this altered content new keywords are extracted. The use of the Entity-free Dataset is always together with the associated keywords and Entities.

4.2.3 Cleaned Dataset. The Cleaned Dataset is made from the Entity-Cleaned Dataset and deletes the title and the artist-name extra from the review_content. This Dataset thus has a review_content where the title, the artist name and the entities are removed. From this new review_content the keywords are extracted again and used within this Dataset.

4.2.4 Basic Schema. The basic schema has no self-defined fields and just uses the ClassicTokenizerFactory as tokenizer, which uses whitespace and punctuation as delimiters [17]. This schema can in fact be seen as using "no schema" and has no extra functionality.

4.2.5 Extended Schema. The extended schema uses three own-defined field types; standard_text, single_words and free_text. The standard_text is used for the fields which information is concrete and searching on those will contain the exact word. For example if you search for a title, you will search for the exact word in the title. The free_text is used for the review_content and has the purpose to elegantly index large blocks of text. The single_words have as goal to tokenize the entities and the keywords. These words are viewed as important in the review_content and with this it is important to find the root of the word and make the word as useful as possible for eventual querying. For example drumming is changed to 'drum' token. An overview of the schema fields can be seen in table 5. Figure 6 shows

Type	Field	Indexed	MultiValued
standard_text	title	True	False
	artist	True	False
	author	True	False
	song release year	True	False
	genre	True	False
	record_label	True	False
single_words	keywords	True	True
	organizations	True	True
	places	True	True
	persons	True	True
	facilities	True	True
	law	True	True
	work_of_art	True	True
	events	True	True
	products	True	True
	norp	True	True
free_text	review_content	True	False
float	score	True	False
int	number of entities	True	False
string	reviewid	False	False

Table 5. Extended schema fields.

the tokenizer and filters that are used for each field type of

the extended schema. Standard_text uses the ClassicTokenizerFactory, which splits the text field into tokens, treating whitespace and punctuation as delimiters [17]. Besides this it uses the ASCIIFoldingFilterFactory, the LowerCaseFilterFactory and the TrimFactory as filters. The function of these is respectively to change alphabetic, numeric, and symbolic Unicode characters which are not in the Basic Latin Unicode block to their ASCII equivalents [18], to convert any uppercase letters in a token to the equivalent lowercase token [19] and to trim leading and/or trailing whitespace from tokens [20].
Single_words uses the KeywordTokenizerFactory as tokenizer, since it is used for keywords and entities. This tokenizer treats the entire text field as a single token [21]. As filters it uses the three filters from standard_text complemented by the EnglishMinimalStemFilterFactory, the EnglishPossessiveFilterFactory and the KStemFilterFactory. The EnglishMinimalStemFilterFactory and the EnglishPossessiveFilterFactory, respectively changes the plural English words to their singular form [21] and removes singular possessives from words [22]. At last the KStemFilterFactory reduces inflected words to their word stem [23].
At last free_text also uses the ClassicTokenizerFactory as tokenizer and all the filters discussed in the single_words field, together with the HyphenatedWordsFilterFactory and the StopFilterFactory. The HyphenatedWordsFilterFactory reconstructs hyphenated words that have been tokenized as two tokens because of a line break or other intervening whitespace in the field test [24] and the StopFilterFactory discards the analysis of, tokens that are on the given stop words list [25]. In this case the general stop words of the English language are used.

standard_text	Tokenizer
	solr.ClassicTokenizerFactory
	Filters
	solr.ASCIIFoldingFilterFactory solr.LowerCaseFilterFactory solr.TrimFilterFactory
single_words	Tokenizer
	solr.KeywordTokenizerFactory
	Filters
	solr.ASCIIFoldingFilterFactory solr.LowerCaseFilterFactory solr.EnglishMinimalStemFilterFactory solr.EnglishPossessiveFilterFactory solr.HyphenatedWordsFilterFactory solr.KStemFilterFactory solr.TrimFilterFactory
free_text	Tokenizer
	solr.ClassicTokenizerFactory
	Filters
	solr.ASCIIFoldingFilterFactory solr.LowerCaseFilterFactory solr.EnglishMinimalStemFilterFactory solr.EnglishPossessiveFilterFactory solr.KStemFilterFactory solr.TrimFilterFactory solr.StopFilterFactory

Table 6. Extended schema field types.

4.2.6 Synonyms schema. The synonyms schema is an extension to the extended schema, however a new filter was introduced called 'solr.SynonymGraphFilterFactory' like previously mentioned, which enables us to apply synonyms to 'keywords', in order to this, a new field type called 'keywords_type' was added, which contained all the tokenizers and filters from 'single_words' together with this new filter, and then assigned to field 'keywords', as presented in the following table 7 and table 8.

4.3 Information needs

4.3.1 Query 1: I want to find albums that are reviewed as boring.

The first query tries to answer the second information need, described in section 2.6. The query, as seen below, uses The Extended DisMax Query Parser [16] and searches for the word boring in the review_content and in the keywords. Where its presence is boosted by 50 in the keywords and 20 in the review_content. Besides this an extra boost is implemented, in the boost-field, that incorporates the score of a review. Reviews with a lower score are favored over the ones with a higher score, since boring albums don't get a high score in general.

q:
boring
q.op:
AND
qf:
review_contentˆ20
keywordsˆ50
boost:
abs(sum(-10, score))

Obtained Results. As seen in table 9 system_4 and 5 have the highest average precision of 94%, followed by system_3, which has an average precision of 92%. System_6 has the lowest average precision of 52%. System_1 has a precision of 76% and system_2 achieved 76% precision. When looking at table 10, the same trends come to the front. System_5 and system_4 both have a precision of 100% at 5 and 10 retrievals and obtain a 84% precision at 25 retrievals. This is also the case for system_3 but this system obtains only a 86% precision at 15, while systems 4 and 5 have a 93%

Type	Field	Indexed	MultiValued
standard_text	title	True	False
	artist	True	False
	author	True	False
	song release year	True	False
	genre	True	False
	record_label	True	False
single_words	organizations	True	True
	places	True	True
	persons	True	True
	facilities	True	True
	law	True	True
	work_of_art	True	True
	events	True	True
	products	True	True
	norp	True	True
free_text	review_content	True	False
float	score	True	False
int	number of entities	True	False
keywords_type	keywords	True	False
string	reviewid	False	False

Table 7. Synonym Schema Fields.

	Tokenizer
standard_text	solr.ClassicTokenizerFactory
	Filters
	solr.ASCIIFoldingFilterFactory
	...
	Tokenizer
	solr.KeywordTokenizerFactory
	Filters
single_words	solr.ASCIIFoldingFilterFactory
	...
	Tokenizer
	solr.KeywordTokenizerFactory
	Filters
keywords_type	solr.SynonymGraphFilterFactory
	solr.ASCIIFoldingFilterFactory
	solr.LowerCaseFilterFactory
	solr.EnglishMinimalStemFilterFactory
	solr.EnglishPossessiveFilterFactory
	solr.HyphenatedWordsFilterFactory
	solr.KStemFilterFactory
	solr.TrimFilterFactory
	Tokenizer
	solr.ClassicTokenizerFactory
	Filters
free_text	solr.ASCIIFoldingFilterFactory
	...

Table 8. Synonym Schema field types.

precision at that point. System_6 has at all points of retrieval the lowest retrieval, followed by system_2. System_1 has until 15 retrievals a 80% precision, after which it increases until 85% with 25 retrievals.
Figure 18 shows the precision-recall curves for the different systems and this query. All these results are obtained by evaluating the 25 first retrieved documents.

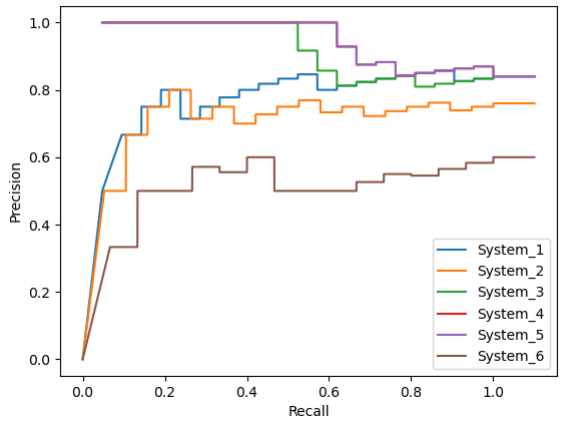


Figure 18. Precision-recall curve of the results for the different systems.

	Average Precision	Recall
system_1	0.762023	1.0
system_2	0.708548	1.0
system_3	0.917173	1.0
system_4	0.9444	1.0
system_5	0.9444	1.0
system_6	0.518665	1.0

Table 9. Average precision and Recall of the retrieval.

	P@05	P@10	P@15	P@20	P@25
system_1	0.8	0.8	0.8	0.85	0.84
system_2	0.8	0.7	0.733333	0.75	0.76
system_3	1.0	1.0	0.866667	0.85	0.84
system_4	1.0	1.0	0.933333	0.85	0.84
system_5	1.0	1.0	0.933333	0.85	0.84
system_6	0.6	0.6	0.533333	0.55	0.6

Table 10. Precision at 5, 10, 15, 20 and 25 retrievals.

Evaluation. Both system_4 as system_5 achieve the best results, both in average precision as the precision at different point of retrieval. As can be seen in figure 19 both system_4 and system_5 have the same precision-recall curves. This is

because the only difference between those systems is that in system_5 the title and the artist name are removed from the review_content. This does not make a difference when searching for the word boring.

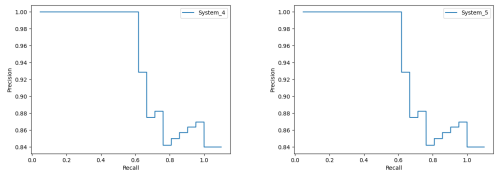


Figure 19. Precision-recall curves of systems 2 and 3.

System_6 has such a low precision, since the synonyms are not completely relevant, for example "oil" and "tone" are synonyms. Oil will not have a big influence, since it is not commonly used together with music reviews, but tone is and this can change the results drastically. System_3 also shows good results, but less then System_4 and 5, since in 3 the entities are not yet removed from the review_content. One entity is for example "Boredoms" and this is refactored to boring when querying, which influences the results. System_1 is better than system_2, which can also be explained through "Boredoms". In system_2 this word will higher the score of a document, while in system_1 only the word "boring" will influence the results and "Boredoms" is not taken into account.
The fact that the systems 4 and 5 have a better result than system_1, with querying over the exact word, shows that by using the right filters and processing the data right eliminations of bad documents can be made.
Also by comparing with the results in section 3.3 in Milestone 2, shows that the query with a basic schema (or without in Milestone 2) now has better results, which indicates that the query has improved. This also because of the use of the edismax query parser [16]. The results of system_2 and with schema of section 3.3 cannot be compared, since they both use a different query and different schema.

4.3.2 Query 2: I want to find albums that contain a drum solo.

This query tries to answer the information need to find albums that contain a drum solo. The words 'drum' and 'solo' are queried with the q field in the fields keywords and review_content with the operator AND. The fields are boosted, respectively with 50 and 20. Besides this is a boost implemented that boosts with 10 if drum and solo are 5 or less places apart in the review_content. The query is executed in the edismax query parser [16] and can be seen below.

q:

drum solo

q.op:

AND

qf:

review_content^20
keywords^50

pf:

review_content^10

ps:

5

Obtained Results. The first 25 retrievals of the query, discussed above, are evaluated. In table 11 the results can be found. This shows that system_1 has the highest average precision of approximately 83%, followed by systems 4, 5 and 6 with an average precision of 80% and system 2 and 3 have an average precision of 78%. They all have a recall of 100% since per system the relevant results were determined for the 25 first results and not for all retrievals possible. This is because around 1800 results are obtained with this query and determining all relevant results of this query cannot be done by hand during the time of this project.

	Average Precision	Recall
system_1	0.831695	1.0
system_2	0.777176	1.0
system_3	0.777176	1.0
system_4	0.801482	1.0
system_5	0.804954	1.0
system_6	0.804954	1.0

Table 11. Average precision and Recall of the retrieval.

Table 12 shows the precision at respectively 5, 10, 15, 20 and 25 retrievals. For system 2 until 6 this result is always the same, namely 80%. System_1 shows different results with 88% at 25 retrievals.

	P@05	P@10	P@15	P@20	P@25
system_1	0.8	0.9	0.933333	0.95	0.88
system_2	0.8	0.8	0.8	0.8	0.8
system_3	0.8	0.8	0.8	0.8	0.8
system_4	0.8	0.8	0.8	0.8	0.8
system_5	0.8	0.8	0.8	0.8	0.8
system_6	0.8	0.8	0.8	0.8	0.8

Table 12. Precision at 5, 10, 15, 20 and 25 retrievals.

Figure 20 shows the precision-recall curves for all the different systems, for the executed query.

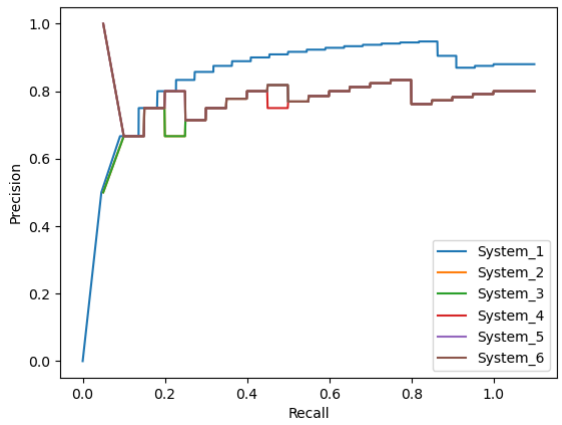


Figure 20. Precision-recall curve of the results for the different systems.

Evaluation. System_1 reaches the best results, with an average precision of 83%. This can also be seen in the precision-recall curves where the precision at recall of 1.0 is higher than for the other systems. However, until a recall of 0.1 is reached it has a lower precision than systems 4, 5 and 6, which indicates that the first retrieval(s) are not relevant. This can be explained because system one does not have an extended schema, so all words are kept as an original token and no keywords are extracted. The retrieval-score for a document is only calculated on the presence of 'drum solo' in the review_content, which will normally yield a good result, since cannot be really misinterpreted. However, only 782 documents are retrieved with system_1 and 1830 with system 4, 5 and 6. This means that a lot of relevant documents will not be retrieved with system_1, while they are retrieved in the other systems.
System 2 and system 3 yield exactly the same results, as can be seen in figure 21. This means that for the original review_content the same results are retrieved, with and without keywords. This is because the keywords are extracted from the review_content and the more a word occurs, the more change it has to be a keyword. If a document possesses the words "drum" and "solo" more in it's review_content, the document will have a higher score in system_2. Besides this, it also has a higher change that "drum" and "solo" occur in its keywords, which again helps to have a higher score in system_3.

Figure 22 shows the precision-recall curves of system_5 and system_6. These are also exactly the same. This is because synonyms are only applied on the keywords and a lot of boosting happens on the review_content, such as the phrase slop. Besides this are not a lot of synonyms used for "drum" and "solo" and these two reasons together give that

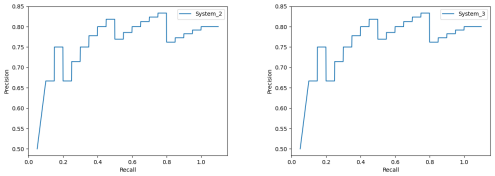


Figure 21. Precision-recall curves of systems 2 and 3.

these systems yield the same curves.

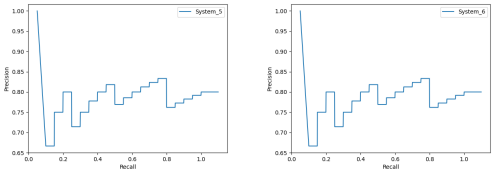


Figure 22. Precision-recall curves of systems 5 and 6.

System_4 is different from all other systems, in that it has a lower precision at around 0.5 recall. System_2 and system_3 also have a lower precision than system_5 and system_6 until around a recall of 30%. This shows that altering the review_content by removing the entities, the title and the artist-name has an impact on the performance of the system.

4.3.3 Query 3: I want to find reviews where Bob Dylan is mentioned but is not the artist.

The third query answers the third information need as discussed in section 2.6. It queries for bob dylan in review_content and persons, and does not allow the artist to be "bob dylan". The querying happens on the words bob and dylan separately, but both have to occur. This is because the authors of the review often refer to Bob Dylan as "Bob" or just "Dylan". Boosting happens on persons with 30, because Bob Dylan is a person, which the pipeline of spaCy [11] recognizes. The query can be seen below and is executed with the edismax query parser [16].

q:
bob dylan
AND
(-artist:"bob dylan"^10)

q.op:
AN

qf:
review_content^10 persons^30

Obtained Results. After evaluating the first 25 retrievals, the results in table 13 and 14 are obtained. Here can be seen that system_1 has the highest average precision of 81% , followed by system_2 with 80% and system_4, 5 and 6 with

average precision of 77,9%. System_3 has the lowest average precision, with a value of 71%. However, in table 14 can be seen that the precision at 25 retrievals is the highest for systems 4, 5 and 6, with 88%. System_1 has only 84% precision at 25 retrievals.

	Average Precision	Recall
system_1	0.805948	1.0
system_2	0.779538	1.0
system_3	0.708603	1.0
system_4	0.778518	1.0
system_5	0.778518	1.0
system_6	0.778518	1.0

Table 13. Average precision and Recall of the retrieval.

	P@05	P@10	P@15	P@20	P@25
system_1	0.8	0.8	0.8	0.85	0.8
system_2	0.8	0.7	0.733333	0.8	0.84
system_3	0.6	0.7	0.666667	0.7	0.76
system_4	0.6	0.8	0.8	0.85	0.88
system_5	0.6	0.8	0.8	0.85	0.88
system_6	0.6	0.8	0.8	0.85	0.88

Table 14. Precision at 5, 10, 15, 20 and 25 retrievals.

Figure 23 shows the precision-recall curves for the different systems and the proposed query.

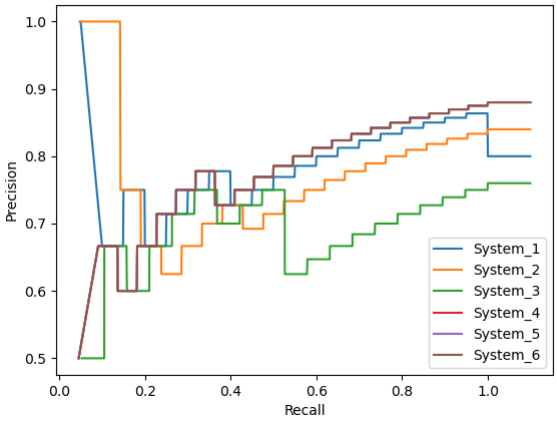


Figure 23. Precision-recall curve of the results for the different systems.

Evaluation. System_2 shows the best results until a recall of 0.2, followed by system_1. This is because they only look at the quantity of occurrences of "Bob" and "Dylan" in the review_content. The better performance of system_2 compared to system_1 also shows that the use of the extended schema has advantages. With a recall higher of 20% system_4, 5 and 6 yield better results. This shows that the use of entities, together with deleting them from the review_content optimises the system. Only when the spaCy pipeline lists the words "Bob" and "Dylan" as the person "Bob Dylan", the words will be listed in the person-entities list and the documents will be retrieved. This gives a higher certainty the result is correct. The low precision in the beginning can be explained by that the album was also of "Bob Dylan". The artist name was listed as "Various Artists" and one of them was "Bob Dylan" and he was talked about a lot, which put him multiple times in the entities-persons list and gave this document a high score. This is also the reason for the bad performance of system_3 since this system boosts the number of occurrences of "Bob Dylan" twice, once when he occurred in the review_content and once for the entities-persons.
Figure 24 shows the precision-recall curves for the systems 4, 5 and 6 separately. Here can be seen that they all have the same curves. This can be explained, since the query boosts on the entities and in all 3 systems, the entities stay the same and they are removed from the review_content. This makes that the same score is generated for the document based on the entities and that also the little altering (only the title and artist name) in the review_content between system_4 and systems 5 and 6 has no influence on the generated score.

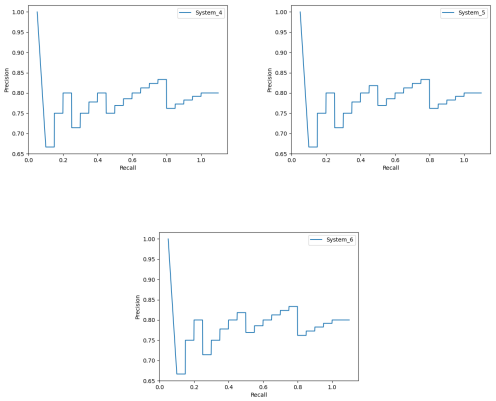


Figure 24. Precision-recall curves of systems 4, 5 and 6.

4.3.4 Query 4: I want to find reviews about albums that include jazz and piano.

For this query we try to look for albums which include jazz and piano. To perform this, we gave more importance to the jazz genres, and then tried to find the word 'piano' in 'review_content' and 'keywords'. Below is the query that we have generated for this information need.

q:

("jazz piano" 20)^2 piano jazz

q.op:

OR

qf:

review_content genre^3 keywords

Obtained Results.

The first 25 retrievals of the query, discussed above, are evaluated. The results obtained can be visualized below:

Table 15 shows the precision at respectively 5, 10, 15, 20 and 25 retrievals.

	P@05	P@10	P@15	P@20	P@25
system_1	0.8	0.8	0.8	0.8	0.76
system_2	1.0	0.9	0.867	0.85	0.84
system_3	0.6	0.8	0.8	0.8	0.8
system_4	1.0	0.9	0.933	0.9	0.92
system_5	1.0	0.9	0.933	0.9	0.92
system_6	1.0	1.0	0.933	0.95	0.92

Table 15. Precision at 5, 10, 15, 20 and 25 retrievals.

Table 16 shows the Average Precision and Recall in different systems

	Average Precision	Recall
system_1	0.78632	0.95
system_2	0.904712	1.0
system_3	0.778902	1.0
system_4	0.934236	1.0
system_5	0.934236	1.0
system_6	0.964543	1.0

Table 16. Average precision and Recall of the retrieval.

Figure 25 shows the precision-recall curves for all the different systems, for the executed query.

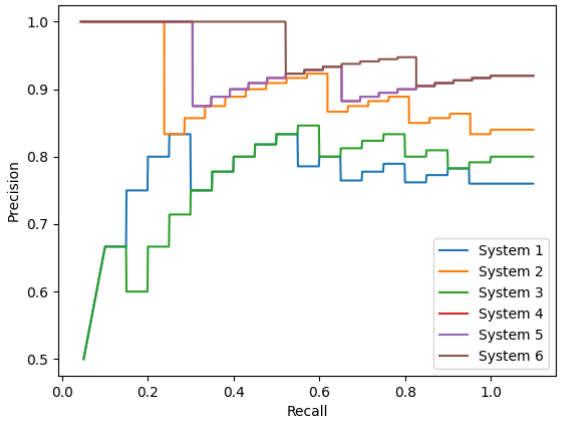


Figure 25. Precision-recall curve of the results for the different systems.

Evaluation.

From the obtained results, we can see that overall system_6 performed the best for this information need, with system_5 and system_4 right after it with a difference of 0.03 for average precision. By looking at system_3, we can see that it had drastically decreased in average precision, which was caused by introducing keywords, it is hard to explain reasoning behind it, but once the entities and keywords were removed from the dataset, gave a boost in average precision which led to a value slighter higher than the previous system that did not contain keywords (system_2), therefore we think that the search system looked for the results which contain more words with "piano" and "jazz" and in return this score gave more importance than genre being jazz, which made results that were not jazz genre to stand out more than the ones that are of this genre, as system_3 began with lower precision at 5 retrievals and then rising to 0.8 and maintaining at this value, is because the first 5 results contained results which were "piano" and "jazz" dense in word-ling and were not actually jazz genre. Additionally, something that we can improve on this information need is to give a higher boost to jazz genre, or even filter out this genre itself and look for "piano" in the context. To conclude this system, overall performed really well, an explanation to this is because normally is common for piano to appear in jazz music.

4.3.5 Query 5: I want to find albums that contain songs which are suitable for studying.

For this query, Lucene Query Parser was used to search for album reviews that potentially contain songs which are suitable studying, just like the one used in Milestone 2, but with 'keywords' were also looked into. As mentioned before, songs which are suitable for studying depends a lot of a person preference, but in our case, we considered these songs to be relaxing and quiet, normally classical and jazz music

are also considered to help us stay focused, and that is why we also give some priority for these genres, however, since the types of genres we have do not contain 'classical music' (analysed using 'OpenRefine' text facets), instead, it was searched inside of 'review_content' and 'keywords' instead of 'genre'. Below is the query that we have generated for this information need.

q:

genre:jazz^2
review_content: calm
review_content: quiet
review_content: ambient
review_content: relaxing
review_content: "classical music"
!review_content: "violent"
keywords: calm^1.5
keywords: quiet^1.5
keywords: relaxing^1.5
keywords: "classical music"1.5

q.op:

OR

fq:

rating_score: [8.0 TO 10.0]

Obtained Results. The first 25 retrievals of the query, discussed above, are evaluated. The results obtained can be visualized below:

Table 17 shows the precision at respectively 5, 10, 15, 20 and 25 retrievals.

	P@05	P@10	P@15	P@20	P@25
system_1	0.8	0.4	0.33	0.25	0.2
system_2	0.8	0.5	0.4	0.35	0.32
system_3	0.4	0.4	0.33	0.3	0.24
system_4	0.6	0.6	0.53	0.55	0.56
system_5	0.6	0.6	0.53	0.55	0.6
system_6	0.6	0.7	0.67	0.55	0.64

Table 17. Precision at 5, 10, 15, 20 and 25 retrievals.

Table 18 shows the Average Precision and Recall in different systems

	Average Precision	Recall
system_1	0.397256	0.714286
system_2	0.470069	0.727273
system_3	0.386765	0.5
system_4	0.570678	1.0
system_5	0.572572	1.0
system_6	0.679862	1.0

Table 18. Average precision and Recall of the retrieval.

Figure 26 shows the precision-recall curves for all the different systems, for the executed query.

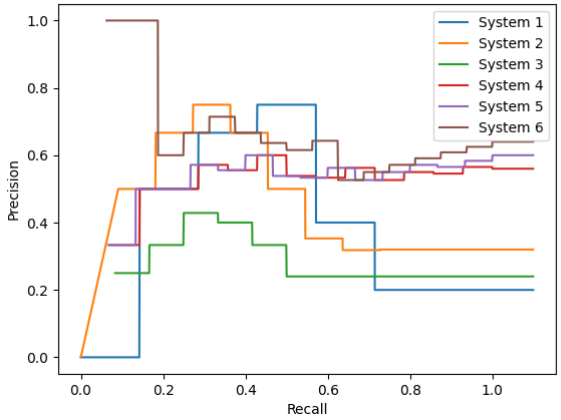


Figure 26. Precision-recall curve of the results for the different systems.

Evaluation. After analysing the results obtained, we can see that system_6 had the best results and system_3 had the worst. The bad performance of system_3 can be explained by the same reason as the one we saw in the previous information, in this case, jazz genre songs tend to have a greater change for a music being suitable for studying, however with introduction of keywords, reviews that contain a lot of "calm", "quiet", "relaxing" words start to appear, and most them can be referring not to the album itself and therefore these will stood up more than genre 'jazz'.

4.3.6 Query 6: I want to find reviews where the author praises the artist's voice.

For this query, Lucene Query Parser was again used to search for reviews where the author praises the artist's voice. For this query a similar strategy was used like the previous query where we looked for songs suitable for studying, where we try to find sentences and words inside the 'review-content', in this case we want to look for words like 'voice', 'vocal', 'beautiful', but at the same time we want to avoid words such as 'terrible', 'bad'. These words are also looked up inside of 'keywords', where we hope it will also work with the keywords synonyms in system 6, so we can get other results with synonym words. Below is the query that we have generated for this information need.

q:

review_content: ("I love voice" 5)^5
review_content: ("unique voice" 5)^5
review_content: "voice"^2
review_content: "vocal"^2
review_content: "I love"
review_content: "great"
review_content: "beautiful"
!review_content: "terrible"
!review_content: "bad"
!keywords: "terrible"
!keywords: "bad"
keywords: voice^4
keywords: unique^2
keywords: perfect^2

q.op:

OR

Obtained Results.

The first 25 retrievals of the query, discussed above, are evaluated. The results obtained can be visualized below:

Table 19 shows the precision at respectively 5, 10, 15, 20 and 25 retrievals.

	P@05	P@10	P@15	P@20	P@25
system_1	0.6	0.5	0.533	0.5	0.48
system_2	0.6	0.5	0.467	0.45	0.52
system_3	0.8	0.5	0.6	0.6	0.6
system_4	0.6	0.5	0.6	0.65	0.6
system_5	0.6	0.5	0.6	0.65	0.6
system_6	0.4	0.4	0.33	0.4	0.4

Table 19. Precision at 5, 10, 15, 20 and 25 retrievals.

Table 20 shows the Average Precision and Recall in different systems

	Average Precision	Recall
system_1	0.580492	1.0
system_2	0.537957	1.0
system_3	0.652069	1.0
system_4	0.641053	1.0
system_5	0.641053	1.0
system_6	0.383002	1.0

Table 20. Average precision and Recall of the retrieval.

Figure 27 shows the precision-recall curves for all the different systems, for the executed query.

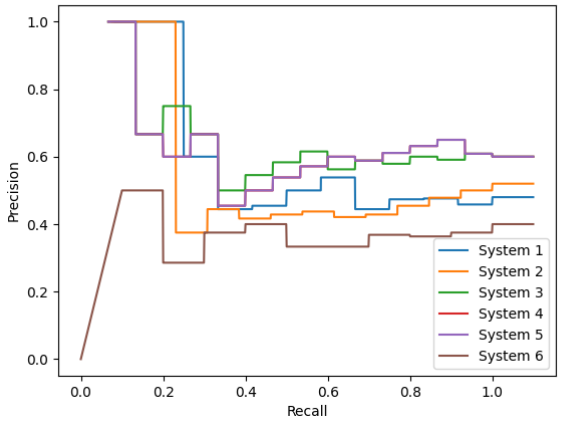


Figure 27. Precision-recall curve of the results for the different systems.

Evaluation.

From the results obtained, it is not very clear which one had the best results, since most of them had similar results, however, it is easy to point out which one is the worst one, which was system_6. System 6 performed the worst compared to the other systems, which was something than was not expected to happen, since it had the synonyms, we expected it to perform better than the others. After further analysing our synonyms file, we found out that the synonyms that we got for the keywords 'voice', 'unique' were not really accurate (were synonyms for other contexts), and therefore these keyword synonyms gave misleading directions to search system, and therefore producing bad results. However, there is a solution to fix this problem, which is to look at the meaning of a word at a giving context and then look for the synonyms, which something hard to do, and therefore we were not able to do this for this Milestone.

Figure 28 shows the synonyms keywords produced by the word 'voice'.

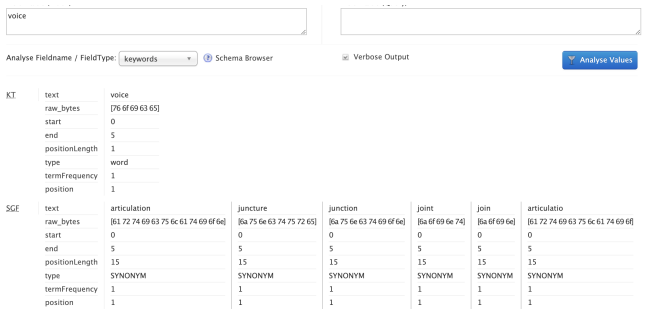


Figure 28. Misleading synonyms produced the the keyword 'voice'.

4.4 General Results

After completing all queries it was seen than on average system_4 and system_5 generated the best precision. This can also be seen in table 21 where the Mean Average Precision (MAP) for every system is calculated. From this we can conclude that the improvements of extracting entities and keywords have an additional positive effect on the document retrieval. Since system_5 and system_4 have the same MAP, the discussion can be made if removing the title and artist name from the review_content has a positive effect. However, in some instances (e.g. query 2) system_5 has the same average precision but the precision at 15 retrievals is better than system_4. With that in mind, the improvements in system_5 are useful especially if later extra information need would be found, which also profit from the removal of the title and artist name. The improvement of adding synonyms didn't have the positive effect as expected, as can be seen in the MAP of system_6. This low precision is due to the illogical use of some synonyms. With a better choice of synonyms this system has the potential to improve its MAP and even become better than system_5.

	Mean Average Precision
system_1	0.693956
system_2	0.696333
system_3	0.703448
system_4	0.778395
system_5	0.779289
system_6	0.688257

Table 21. Mean Average Precision per system.

4.5 Changes to M2

The search tasks in Milestone 2 were still not optimal after the second iteration. We changed these during the third Milestone, so the information needs are more clear and querying could be done in a more optimized way.

5 Conclusion

The first milestone was successfully completed, since a relevant dataset for the project has been selected, prepared and characterised. The quality of the data and authority of the source have been assessed and approved. The data has been analysed and characteristics have been composed. A data processing pipeline was developed and a conceptual model for the data has been established.

For the second milestone, we were able to get familiar with an information retrieval tool (Solr). During this milestone, the dataset resulting from the previous milestone was uploaded to Solr, by testing our information needs using 2 different schemas (one with schema and one without schema), we found out that the use of schema indeed helps us boost the performance to search for our information needs. However only a slight improvement was seen when comparing the results, but a better choice of schema could improve it more. During Milestone 3 extra information needs were added and new improvements were developed. The three big improvements were the extraction of keywords and entities, the use of synonyms and the altering of the review_content to extract keywords. After conducting the experiments with different queries, it was clear that the use of synonyms didn't really improve the system, since the keywords were not relevant. The extraction of entities helps a lot when searching for specific persons, places, etc. But the extraction of keywords has the most influence on improving the system, especially if the entities are removed. Our developed information needs mostly make use of normal word (e.g. "drum", "boring", "peace" and not persons). By removing the entities only the normal words remain, which are then used to extract the keywords. These keywords are then very helpful when querying our needed information.

For future work and improvements, we could imrpove the use of synonyms, so more relevant synonyms are use. Besides this we could also look into better boosting (e.g. a person that occurs more than one time in the entities is only boosted once or boosting depending on the index of the keywords, since more frequent words occur first in the keywords).

References

[1] Idera. Data Growth. https://www.idera.com/glossary/data-growth/

[2] [n.d.]. Pandas. https://pandas.pydata.org/

[3] [n.d.]. Beatiful Soap. https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[4] Nolan Conaway. 18,393 Pitchfork Reviews. https://www.kaggle.com/datasets/nolanbconaway/pitchfork-data

[5] [n.d.]. Matplotlib. https://matplotlib.org/

[6] [n.d.]. Seaborn. https://seaborn.pydata.org/

[7] [n.d.]. OpenRefine. https://openrefine.org/

[8] [n.d.] Solr. https://solr.apache.org/

[9] [n.d] Query Syntax and Parsing. https://solr.apache.org/guide/8_10/query-syntax-and-parsing.html

[10] [n.d.] Python. https://www.python.org/

[11] [n.d.] spaCy. https://spacy.io/

[12] [n.d.] collections. https://docs.python.org/3/library/collections.html

[13] [n.d.] TextBlob Sentiment Analysis. https://textblob.readthedocs.io/en/dev/quickstart.html

[14] [n.d.] Solr Filter-descriptions. https://solr.apache.org/guide/7_0/filter-descriptions.html

[15] [n.d.] Offline database of synonyms/thesaurus. https://github.com/zaibacu/thesaurus

[16] [n.d.] The Extended DisMax Query Parser. https://solr.apache.org/guide/6_6/the-extended-dismax-query-parser.html

[17] [n.d] Standard Tokenizer. https://solr.apache.org/guide/8_10/tokenizers.html
[18] [n.d.] ASCII Folding Filter. https://solr.apache.org/guide/8_10/filter-descriptions.htmlascii-folding-filter
[19] [n.d.] Lower Case Filter. https://solr.apache.org/guide/8_10/filter-descriptions.htmllower-case-filter
[20] [n.d.] Trim Filter. https://solr.apache.org/guide/8_10/filter-descriptions.htmltrim-filter
[21] [n.d.] Keyword Tokenizer. https://solr.apache.org/guide/8_10/tokenizers.htmlkeyword-tokenizer
[22] [n.d.] English Minimal Stem Filter. https://solr.apache.org/guide/8_10/filter-descriptions.htmlenglish-minimal-stem-filter
[22] [n.d.] English Minimal Stem Filter. https://solr.apache.org/guide/8_10/filter-descriptions.htmlenglish-possessive-filter
[23] [n.d.] KStem Filter. https://solr.apache.org/guide/8_10/filter-descriptions.htmlkstem-filter
[24] [n.d.] Hyphenated Words Filter. https://solr.apache.org/guide/8_10/filter-descriptions.htmlhyphenated-words-filter
[25] [n.d.] Stop Filter. https://solr.apache.org/guide/8_10/filter-descriptions.htmlstop-filter
[26] [n.d.] PitchFork. https://pitchfork.com/