UNIVERSIDAD DE GRANADA

Recuperación de Información 2024/25

Practica 6

Implementation of an Information Retrieval System using Lucene – Facetes and classification

Group-name: JuevesC++

- **Carolin Irrgang ,** e-mail: **cirrgang@correo.ugr.es**
- **Christiana Mousele ,** e-mail: am069030@correo.ugr.es

Contents

Overview of our project

In this práctica we reused code of the last prácticas and changed it according to our needs. We reindexed the US reviews documents to suit a taxonomy that will help for faceting and classification in lucene as well as for sentiment analysis. We changed some fields to be different types so that we are able to correctly work with them. Additionally, we created two classes in Java, one being responsible for indexing called Indexer while the other called Facetery is responsible for faceting/classifying and sentiment analysis. Both are accessed through a single main method in the Facetery class.

User's Guide and explanations

To start the program, run the .jar file. It will first ask to put an index and taxonomy directory. Both should be empty, if an index is not present. If files have been correctly indexed, the user can paste the respective directories including the luke indices.

Indexing

The program will then ask to choose an option of the following four:

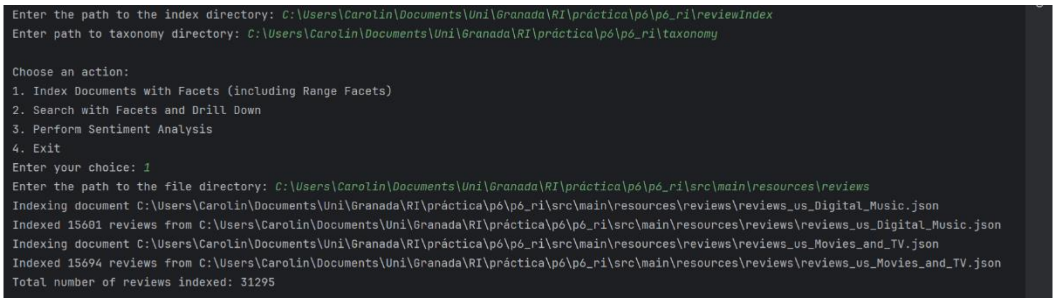
1. Index Documents with Facets (including Range Facets)

2. Search with Facets and Drill Down

3. Perform Sentiment Analysis

4. Exit

If the given directories are empty, one has to start with putting a 1. The program will then ask to give the directory path to the files to be indexed. In our case these have to be US review JSON files. It'll index everything and create a taxonomy using the Indexer class. We indexed 2 review files to give an example and receive the amount of indexed reviews:



```
Enter the path to the index directory: C:\Users\Carolin\Documents\Uni\Granada\RI\práctica\p6\p6_ri\reviewIndex
Enter path to taxonomy directory: C:\Users\Carolin\Documents\Uni\Granada\RI\práctica\p6\p6_ri\taxonomy

Choose an action:
1. Index Documents with Facets (including Range Facets)
2. Search with Facets and Drill Down
3. Perform Sentiment Analysis
4. Exit
Enter your choice: 1
Enter the path to the file directory: C:\Users\Carolin\Documents\Uni\Granada\RI\práctica\p6\p6_ri\src\main\resources\reviews
Indexing document C:\Users\Carolin\Documents\Uni\Granada\RI\práctica\p6\p6_ri\src\main\resources\reviews\reviews_us_Digital_Music.json
Indexed 15601 reviews from C:\Users\Carolin\Documents\Uni\Granada\RI\práctica\p6\p6_ri\src\main\resources\reviews\reviews_us_Digital_Music.json
Indexing document C:\Users\Carolin\Documents\Uni\Granada\RI\práctica\p6\p6_ri\src\main\resources\reviews\reviews_us_Movies_and_TV.json
Indexed 15694 reviews from C:\Users\Carolin\Documents\Uni\Granada\RI\práctica\p6\p6_ri\src\main\resources\reviews\reviews_us_Movies_and_TV.json
Total number of reviews indexed: 31295
```

Most importantly, the fields "asin","overall" and "cleanreviewtime" are stored as FacetFields where "cleanreviewtime" becomes hierarchical to "date" including the nodes "year" and "month".

Search

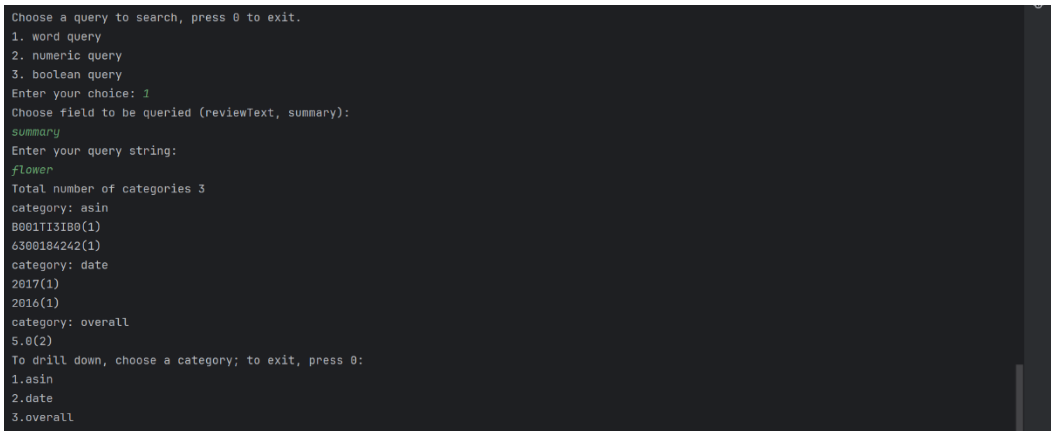
If an index is present, the search can begin. When enterin choice 2, another menu is opened:
1. word query

2. numeric query

3. boolean query which we will be explaining now.

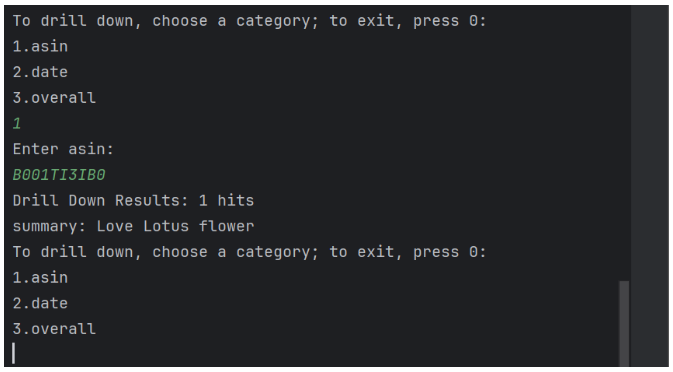
Simple Query

The first option is a simple query, where you can choose a field to search (any is fine as they're all stored, most sense has reviewText or summary). You'll be asked to name the field and then the term you're searching for. We've put the field "summary" and searching for "flower" as an example:



```
Choose a query to search, press 0 to exit.
1. word query
2. numeric query
3. boolean query
Enter your choice: 1
Choose field to be queried (reviewText, summary):
summary
Enter your query string:
flower
Total number of categories 3
category: asin
B001TI3IB0(1)
6300184242(1)
category: date
2017(1)
2016(1)
category: overall
5.0(2)
To drill down, choose a category; to exit, press 0:
1.asin
2.date
3.overall
```

The amount of hits in each categories are displayed and you're now able to drill down into any category which will then show you the end result. For example for the first asin:

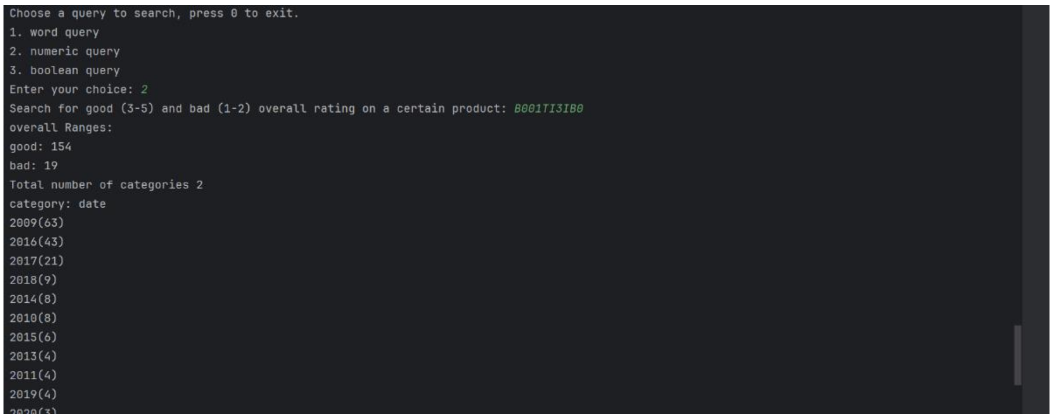


```
To drill down, choose a category; to exit, press 0:
1.asin
2.date
3.overall
1
Enter asin:
B001TI3IB0
Drill Down Results: 1 hits
summary: Love Lotus flower
To drill down, choose a category; to exit, press 0:
1.asin
2.date
3.overall
```

You can exit by pressing 0.

Numeric Query

This second option lets you see how many positive and negative ratings a certain product has. For this you need the asin. We'll take the past asin as an example:

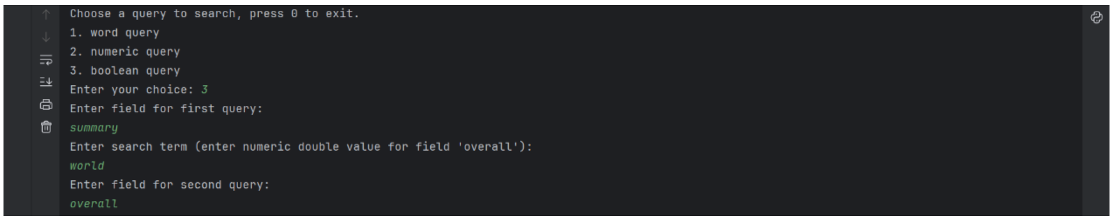


```
Choose a query to search, press 0 to exit.
1. word query
2. numeric query
3. boolean query
Enter your choice: 2
Search for good (3-5) and bad (1-2) overall rating on a certain product: B001TI3IB0
overall Ranges:
good: 154
bad: 19
Total number of categories 2
category: date
2009(63)
2016(43)
2017(21)
2018(9)
2014(8)
2010(8)
2015(6)
2013(4)
2011(4)
2019(4)
2020(3)
```

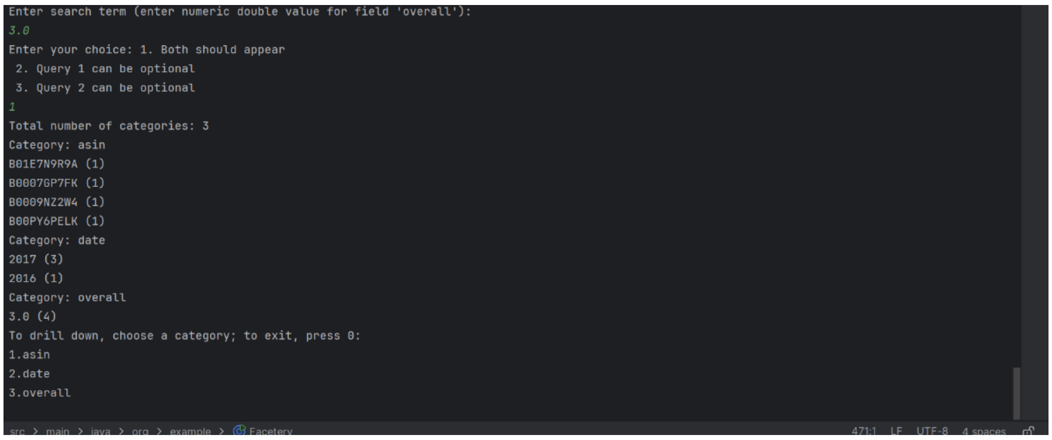
It shows the amount of good and bad overall's as well as an overview of the counts in the categories.

Boolean Query

The third option lets you pick which fields should be considered and which term in each field. You can then decide if both or just one field has to appear. A Boolean query will be carried out.



```
Choose a query to search, press 0 to exit.
1. word query
2. numeric query
3. boolean query
Enter your choice: 3
Enter field for first query:
summary
Enter search term (enter numeric double value for field 'overall'):
world
Enter field for second query:
overall
```



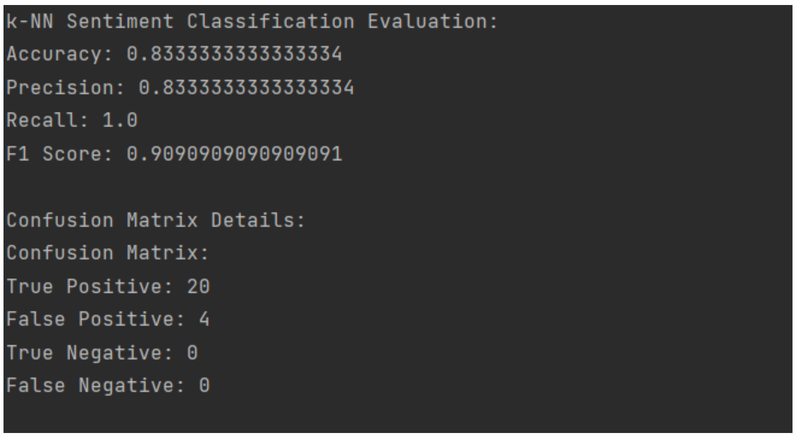
```
Enter search term (enter numeric double value for field 'overall'):
3.0
Enter your choice: 1. Both should appear
 2. Query 1 can be optional
 3. Query 2 can be optional
1
Total number of categories: 3
Category: asin
B01E7N9R9A (1)
B0007GP7FK (1)
B0009NZ2W4 (1)
B00PY6PELK (1)
Category: date
2017 (3)
2016 (1)
Category: overall
3.0 (4)
To drill down, choose a category; to exit, press 0:
1.asin
2.date
3.overall
```

Now you can drill down the categories like in the first simple query.

Sentiment analysis

We created an index with the reviews and the sentiment field added only for this class. In this class we perform sentiment analysis on text reviews using two machine learning algorithms: k-Nearest Neighbors (k-NN) and Naive Bayes. It processes pre-indexed documents stored in a Lucene index (specified by INDEX_PATH) and splits them into training and testing datasets using prepareDatasets. The application predicts the sentiment of a review by calculating similarity (for k-NN) or probabilities (for Naive Bayes). For k-NN, the similarity between documents is computed using cosine similarity, as shown in the calculateTextSimilarity method:

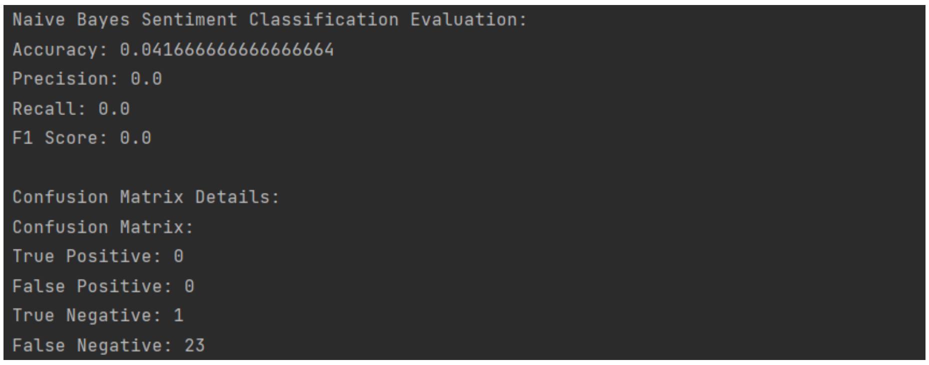
```
double dotProduct = 0.0, magnitude1 = 0.0, magnitude2 = 0.0;
for (String term : tf1.keySet()) {
  dotProduct += tf1.get(term) * tf2.getOrDefault(term, 0);
  magnitude1 += Math.pow(tf1.get(term), 2);
}
return dotProduct / (Math.sqrt(magnitude1) * Math.sqrt(magnitude2));
```



```
k-NN Sentiment Classification Evaluation:
Accuracy: 0.8333333333333334
Precision: 0.8333333333333334
Recall: 1.0
F1 Score: 0.9090909090909091

Confusion Matrix Details:
Confusion Matrix:
True Positive: 20
False Positive: 4
True Negative: 0
False Negative: 0
```

Naive Bayes predicts sentiment by calculating the likelihood of each word belonging to a positive or negative class, incorporating Laplace smoothing:

```
double wordProbPositive = (positiveWordCount.getOrDefault(word, 0) + 1) /
(totalPositiveWords + positiveWordCount.size());
positiveProbability += Math.log(wordProbPositive);
```

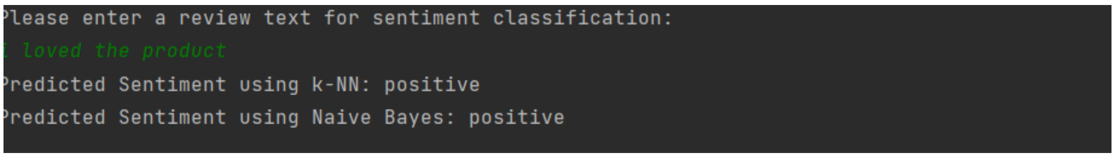


```
Naive Bayes Sentiment Classification Evaluation:
Accuracy: 0.041666666666666664
Precision: 0.0
Recall: 0.0
F1 Score: 0.0

Confusion Matrix Details:
Confusion Matrix:
True Positive: 0
False Positive: 0
True Negative: 1
False Negative: 23
```

This is not working correctly here because the model is almost entirely failing to identify positive sentiment. It only correctly identifies a single negative review (True Negative = 1), but it misses all the positive reviews (False Negatives = 23) and doesn't predict any positive reviews correctly.

The application evaluates both algorithms using a Confusion Matrix, providing metrics such as accuracy, precision, recall, and F1 score. Users can input text to classify its sentiment interactively, with predictions displayed for both classifiers. For example:

```
System.out.println("Predicted Sentiment using k-NN: " + predictedSentimentKNN);

System.out.println("Predicted Sentiment using Naive Bayes: " + predictedSentimentNB);
```



```
Please enter a review text for sentiment classification:
I loved the product
Predicted Sentiment using k-NN: positive
Predicted Sentiment using Naive Bayes: positive
```

The problem with this part of the code is that the predictions are not always right , so the code is not always very functional.

Groupwork

Like in the past exercises, Christiana and Carolin worked closely together sharing the work. In this exercise we shared the work into queries/ faceting and sentiment analysis. Christiana laid the groundworks in the Facetery class by designing the console output and first methods and continued to work on the sentiment analysis. Simultaneously, Carolin tweaked the Indexer class to adjust it to the new taxonomy need. She then continued to work in the Facetery class, adding content to option number 2: Search with Facets and Drill Down. We both were responsible for our respective documentation.