Data Mining for Business and Society - Homework 1

Federico Zarfati id. 1227399

April 15 2016

Instructions

This is the report as requested for the homework subsmission. To replicate every step of the process one has to:

1) Generate all the output files using the scripts provided (changing the paths, the scripts generate files that should be moved in order to make the process work, e.g. the collection has to be moved inside a folder containing the index)
2) Move all the output files generated in a common folder with the python and bash scripts
3) Launch the bash scripts. FIRST performance_cran.sh THEN performance_time.sh (one per dataset). This will generate the Report.csv file. This file is already provided with the zip folder
4) Launch the python software for the plots (one can comment the lines of the script to generate only one plot)

Objective and simple statistics

The objective is to find the best configuration (in terms of stemming method and scorer function) for the search engine, using the available Ground-Truth data. The two dataset used are named Cranfield and Time. The Cranfield dataset is composed of 1400 html documents that we can retrieve using the queries provided (225 queries). The Time dataset is composed of 423 html documents and the queries provided are 83.

Analyzing the files provided we can easly notice that the ground truth files (the files that show us the relevant documents for every query) do not comprehend all the query. Given that, I managed to write an evaluation software that takes into account this issue.

Stemmers and scorer functions

As stated in the homework specifications, the stemmers used are: the default stemmer, the Stemmer for the English language and the Stemmer for the English language able to filter stopwords. Furthermore, the scorer functions used are: NumOccurrences, a trivial scorer function that computes the score by adding the number of occurrences within the current document of each query term, TF/IDF, a well known scorer function, and BM25.

Scripts for the Cranfield dataset

Attention: In this report we will provide all the scripts used in order to create the collection, indexes and run all the queries. The scripts provided can not be runned on different machines without changing the paths. Furthermore, for some scripts, the files generated were moved to meet the requirements of the scripts' options. For example, the collection files were moved in the indexes' folders in order to run the queries' scripts.

In order to evaluate the performance of the various Stemmer-Scorer combinations we have to create first the collection with the following script:

```
find Cranfield_DATASET/cran -iname \*.html |
java it.unimi.di.big.mg4j.document.FileSetDocumentCollection
-f HtmlDocumentFactory -p encoding=UTF-8 Cran_DATASET_DATA.collection
```

Cranfield - Default Stemmer and Scorers

Now that the collection file is generated we can build the first index with the Default Stemmer

```
#Default Stemmer
java it.unimi.di.big.mg4j.tool.IndexBuilder --downcase -S
cran_workspace/Cran_DATASET_DATA.collection cran_workspace/defaultStemmer/Cran_DATASET_DATA
```

Now we can compute the queries for every scorer function and obtain the output files

```
#CountScorer
java homework.RunAllQueries_HW "cran_workspace/defaultStemmer/Cran_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Cranfield_DATASET/cran_all_queries.tsv
"CountScorer" "1:2" cran_workspace/output_cran_default_CountScorer.tsv
```

```
#TFiDFScorer
java homework.RunAllQueries_HW "cran_workspace/defaultStemmer/Cran_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Cranfield_DATASET/cran_all_queries.tsv
"TfIdfScorer" "1:2" cran_workspace/output_cran_default_tfidf.tsv
```

```
#BM25Scorer
java homework.RunAllQueries_HW "cran_workspace/defaultStemmer/Cran_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Cranfield_DATASET/cran_all_queries.tsv
"BM25Scorer" "1:2" cran_workspace/output_cran_default_BM25Scorer.tsv
```

Cranfield - English Stemmer and Scorers

Using the same collection file we can build the second index with the English Stemmer

```
#English Stemmer
java it.unimi.di.big.mg4j.tool.IndexBuilder
-t it.unimi.di.big.mg4j.index.snowball.EnglishStemmer -S
cran_workspace/englishStemmer/Cran_DATASET_DATA.collection
cran_workspace/englishStemmer/Cran_DATASET_DATA
```

Now we can compute the queries for every scorer function and obtain the output files

```
#CountScorer
java homework.RunAllQueries_HW "cran_workspace/englishStemmer/Cran_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Cranfield_DATASET/cran_all_queries.tsv
"CountScorer" "1:2" cran_workspace/englishStemmer/output_cran_english_CountScorer.tsv
```

```
#TFiDFScorer
java homework.RunAllQueries_HW "cran_workspace/englishStemmer/Cran_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Cranfield_DATASET/cran_all_queries.tsv
"TfIdfScorer" "1:2" cran_workspace/englishStemmer/output_cran_english_tfidf.tsv
```

```
#BM25Scorer
java homework.RunAllQueries_HW "cran_workspace/englishStemmer/Cran_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Cranfield_DATASET/cran_all_queries.tsv
"BM25Scorer" "1:2" cran_workspace/englishStemmer/output_cran_english_BM25Scorer.tsv
```

Cranfield - English Stopwords Stemmer and Scorers

Using the same collection file we can build the third index with the English Stopwords Stemmer

```
#Stopwords Stemmer
java it.unimi.di.big.mg4j.tool.IndexBuilder -t homework.EnglishStemmerStopwords -S
cran_workspace/stopStemmer/Cran_DATASET_DATA.collection
cran_workspace/stopStemmer/Cran_DATASET_DATA
```

Now we can compute the queries for every scorer function and obtain the output files

```
#CountScorer
java homework.RunAllQueries_HW "cran_workspace/stopStemmer/Cran_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Cranfield_DATASET/cran_all_queries.tsv
"CountScorer" "1:2" cran_workspace/stopStemmer/output_cran_stop_CountScorer.tsv
```

```
#TFiDFScorer
java homework.RunAllQueries_HW "cran_workspace/stopStemmer/Cran_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Cranfield_DATASET/cran_all_queries.tsv
"TfIdfScorer" "1:2" cran_workspace/stopStemmer/output_cran_stop_tfidf.tsv
```

```
#BM25Scorer
java homework.RunAllQueries_HW "cran_workspace/stopStemmer/Cran_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Cranfield_DATASET/cran_all_queries.tsv
"BM25Scorer" "1:2" cran_workspace/stopStemmer/output_cran_stop_BM25Scorer.tsv
```

Scripts for the Time dataset

Attention: In this report we will provide all the scripts used in order to create the collection, indexes and run all the queries. The scripts provided can not be runned on different machines without changing the paths. Furthermore, for some scripts, the files generated were moved to meet the requirements of the scripts' options. For example, the collection files were moved in the indexes' folders in order to run the queries' scripts.

In order to evaluate the performance of the various Stemmer-Scorer combinations we have to create first the collection with the following script:

```
find Time_DATASET/time -iname \*.html |
java it.unimi.di.big.mg4j.document.FileSetDocumentCollection
-f HtmlDocumentFactory -p encoding=UTF-8 Time_DATASET_DATA.collection
```

Time - Default Stemmer and Scorers

Now that the collection file is generated we can build the first index with the Default Stemmer

```
#Default Stemmer
    java it.unimi.di.big.mg4j.tool.IndexBuilder --downcase -S
    time_workspace/Time_DATASET_DATA.collection
    time_workspace/indexes/default/Time_DATASET_DATA
```

Now we can compute the queries for every scorer function and obtain the output files

```
#CountScorer
java homework.RunAllQueries_HW
"time_workspace/default_CountScorer/Time_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Time_DATASET/time_all_queries.tsv
"CountScorer" "1:2" time_workspace/output_time_default_CountScorer.tsv
```

```
#TFiDFScorer
java homework.RunAllQueries_HW
"time_workspace/default_CountScorer/Time_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Time_DATASET/time_all_queries.tsv
"TfIdfScorer" "1:2" time_workspace/output_time_default_tfidf.tsv
```

```
#BM25Scorer
java homework.RunAllQueries_HW "time_workspace/default_CountScorer/Time_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Time_DATASET/time_all_queries.tsv
"BM25Scorer" "1:2" time_workspace/output_time_default_BM25Scorer.tsv
```

Time - English Stemmer and Scorers

Using the same collection file we can build the second index with the English Stemmer

```
#English Stemmer
java it.unimi.di.big.mg4j.tool.IndexBuilder
-t it.unimi.di.big.mg4j.index.snowball.EnglishStemmer
-S time_workspace/englishStemmer/Time_DATASET_DATA.collection
time_workspace/englishStemmer/Time_DATASET_DATA
```

Now we can compute the queries for every scorer function and obtain the output files

```
#CountScorer
java homework.RunAllQueries_HW "time_workspace/englishStemmer/Time_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Time_DATASET/time_all_queries.tsv
"CountScorer" "1:2" time_workspace/englishStemmer/output_time_english_CountScorer.tsv
```

```
#TFiDFScorer
java homework.RunAllQueries_HW "time_workspace/englishStemmer/Time_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Time_DATASET/time_all_queries.tsv
"TfIdfScorer" "1:2" time_workspace/englishStemmer/output_time_english_tfidf.tsv
```

```
#BM25Scorer
java homework.RunAllQueries_HW "time_workspace/englishStemmer/Time_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Time_DATASET/time_all_queries.tsv
"BM25Scorer" "1:2" time_workspace/englishStemmer/output_time_english_BM25Scorer.tsv
```

Time - English Stopwords Stemmer and Scorers

Using the same collection file we can build the third index with the English Stopwords Stemmer

```
#Stopwords Stemmer
java it.unimi.di.big.mg4j.tool.IndexBuilder -t homework.EnglishStemmerStopwords
-S time_workspace/stopStemmer/Time_DATASET_DATA.collection
time_workspace/stopStemmer/Time_DATASET_DATA
```

Now we can compute the queries for every scorer function and obtain the output files

```
#CountScorer
java homework.RunAllQueries_HW "time_workspace/stopStemmer/Time_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Time_DATASET/time_all_queries.tsv
"CountScorer" "1:2" time_workspace/stopStemmer/output_time_stop_CountScorer.tsv
```

```
#TFiDFScorer
java homework.RunAllQueries_HW "time_workspace/stopStemmer/Time_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Time_DATASET/time_all_queries.tsv
"TfIdfScorer" "1:2" time_workspace/stopStemmer/output_time_stop_tfidf.tsv
```

```
#BM25Scorer
java homework.RunAllQueries_HW "time_workspace/stopStemmer/Time_DATASET_DATA"
/Users/Federico/Documents/University/Data\ Mining/HW1/Time_DATASET/time_all_queries.tsv
"BM25Scorer" "1:2" time_workspace/stopStemmer/output_time_stop_BM25Scorer.tsv
```

Evaluation of the Performance

Notes on the p@k metric

As required from the homework, it has been developed a Python software that computes the p@k with k=1,3,5,10 for every Stemmer-Scorer combination. However, the p@k formula used to compute the performance metric is different from the one presented in class. In fact, it has been noticed that, given the output of a query and the corresponding ground truth set, a p@k computed dividing by k does not give a valuable information on the performance. This because, dividing by k, the p@k will never be one even if the system retrieved all the relevant documents. To solve this problem the p@k was changed computing the division by hte minimum between k and the cardinality of the set of the relevant documents.

Python software

The following script has been developed in order to compute the p@k given two .tsv file (output of the combination and the groundtruth). This code chunck does not show the full script that is available entirely on the file evaluatePerformance.py provided.

```
def avg_p_at_k(output, ground):
    """Save computed p@k in a Report.csv file

    The function takes two argument: queries and ground truth.
    For every p@k (1,3,5,10) the function compute the p@k
    calculating the intersection between the query's results set and
    the set of relevant documents. It divides the result of the intersection
    by the minimum between k and the length of the ground set.
    Every result is added to the previous and then divided by the number of queries.
    """
    global values
    counter = 0
    total=0

    for k in [1,3,5,10]: # For each k

        for query in ground.keys(): # For each query in groundtruth
            total += len(set(output[query][:k]) & set(ground[query]))*1.0/min(k,len(ground[query]))
            print(k)
            print(set(output[query][:k]))
            counter +=1    # Query counter

        # Total of the k divided by the number of the queries
        values.append((total*1.0)/counter)

        # Reset the counters
        total = 0
        counter = 0

    # Save the data
    with open("Report.csv", "a") as fp:

        wr = csv.writer(fp, dialect='excel')

        wr.writerow(values)

def main():
    global values
    values = []

    #Read the file from the main arguments
    queries = pd.read_csv(sys.argv[1], sep="\t")
    ground  = pd.read_csv(sys.argv[2], sep="\t" )

    #Groupd the tsv file for query_id
    out = queries.groupby('Query_ID')
    tru = ground.groupby('Query_id')
```

```
    #Create dictionaries with query_id as keys
    out_dic = {k: g["Doc_ID"].tolist() for k,g in out}
    tru_dic = {k: g["Relevant_Doc_id"].tolist() for k,g in tru}

    #Compute the Average P@k
    avg_p_at_k(out_dic,tru_dic)


if __name__=="__main__":
    main()
```

Python script for plotting

The following script has been developed in order to plot the Report.csv file. This code chunck does not show the full script that is available entirely on the file plotPerf.py provided.

```
import pandas as pd
import matplotlib.pyplot as plt

report=pd.read_csv('Report.csv', header=None, names=[1,3,5,10])

documents=("Cranfield","Time")
stemmers=(" --downcase", "EnglishStemmer" ,"EnglishStemmerStopwords" )
scorers=("CountScorer" ,"TfIdfScorer", "BM25Scorer" )


cran= report[:9]
time= report[9:]
leg=[]
for i in stemmers:
    for j in scorers:
        leg.append(i+' '+j)

tableau20 = [(31, 119, 180), (174, 199, 232), (255, 127, 14), (255, 187, 120),
             (44, 160, 44), (152, 223, 138), (214, 39, 40), (255, 152, 150),
             (148, 103, 189), (197, 176, 213), (140, 86, 75), (196, 156, 148),
             (227, 119, 194), (247, 182, 210), (127, 127, 127), (199, 199, 199),
             (188, 189, 34), (219, 219, 141), (23, 190, 207), (158, 218, 229)]

# Scale the RGB values to the [0, 1] range, which is the format matplotlib accepts.
for i in range(len(tableau20)):
    r, g, b = tableau20[i]
    tableau20[i] = (r / 255., g / 255., b / 255.)
plt.gca().set_color_cycle(tableau20)


plt.subplot(2, 1, 1)
plt.plot(cran.T)
plt.title('Performance of Stemmer-Scorer Combinations on Cranfield Dataset ')
plt.ylabel('Precision @ K')
plt.xlabel('k')
plt.xticks([1,3,5,10])
plt.legend( tuple(leg),fontsize='xx-small', framealpha=0.5, shadow=True, title="Legend",
```

```
fancybox=True,loc='upper center', ncol=5)
plt.gcf().set_size_inches(20,20)


plt.subplot(2, 1, 2)

plt.plot(time.T)
plt.title('Performance of Stemmer-Scorer Combinations on Time Dataset ')
plt.ylabel('Precision @ K')
plt.xlabel('k')
plt.xticks([1,3,5,10])
plt.gcf().set_size_inches(20,20)
plt.legend( tuple(leg),fontsize='xx-small', framealpha=0.5, shadow=True, title="Legend",
fancybox=True,loc='upper center', ncol=5)



plt.show()
```

Results

As we can see from the plots the best combination of Stemmer-Scorer is the English Stemmer Stopwords with the BM25 scorer for all the datasets. The two plots show us that taken individually the stemmer English Stopwords is the best and we can say the same for the BM25 scorer. We can notice how the BM25 performance influence even the bad default stemmer. Furthermore, as we can see in the Time Dataset plot, the English Stemmer Stopwords has the same effect on poorer scorers, but we have to point out the adjacency of the two combinations: Stopwords-BM25 and English-BM25.

The Plots are avaliable in a better format inside the zip folder.