Report for Project 3 of COMP 479 by Silas Kalinowski ID: 40256077

The demo can be found in the file demo.pdf.

Design and summary of approach:

My project consist of 6 modules:

The first module named 'main.py' calls the needed functions for the time comparison of the naïve indexer and SPIMI inspired indexer, and starts the query.

The module 'module1.py' contains the function 'read_extract'. This function is identical to the read_extract() function of Project 2. It returns the NEWID combined with the article's body in a dictionary. This function is used to construct the index.

The module 'subproject1a.py' contains four functions: the function 'preprocess' creates a subcorpus based on a given token limit, the function 'spimi_inspired' implements the SPIMI inspired indexer, the naïve indexer from Project2 is implemented in the 'naïve_indexer' function. The last function called 'create_index' creates an index based on a sorted List of term-documentID pairs and is used by the 'naïve_indexer' function.

The module 'subproject2b.py' implements the creation of the uncompressed index. It is based on the the create_index function of Project 2 but the index stores more values. The BM25 needs the term frequency of every term for every document and the inverse document frequency for every term. That is why I store these values in the index also. The index has the following structure now: term->[idf, (docID, tf), …].This index is saved as index.txt. Also, another index that store the length of every document, and the average length of the documents is created. This index is saved as index_doct.txt.

The module 'subproject2.py' implements the queries for the Subproject2. It contains 4 functions: the function single_query_processor implements the single word query, the function or_query_processor implements an OR query for a given input, the function and_query_processor implements an AND query for a given input, the function bm25query implements a query which ranks the documents by the BM25 formula, to rank the documents the function tf is used. The function tf returns the term frequency for a given term and document based on an index. Lastly, the function get_query_input implements the query search in console.

The module 'Helper_functions.py' contains six function that are used to read an index from a txt file, write an index into a txt file, write the document index into a txt file and read the document index from a txt and save the results of a query in a txt. The module is based on the module from Project2 but was adapted to the new structure of the indices.

For my design I ensured that every part of the project is implemented in separate modules. The modules then again consist of function that implement a specific part of the assignment.

The index is stored in a dictionary when they are being created. I chose dictionary because lookup and inserting elements is very efficient. Also the to track the term frequency for every term in each document I used a dictionary which made it very efficient.

The query processors can be used with any indices as long as the index was saved in a txt file with the specific format mentioned above. This design enables me to create the index ones and search on the index as often as desired without having to create it again.

I also implemented a function that lets the user select the desired query in the console and search for terms. They can also run the sample and test queries. This function is automatically started when running the main file.

The sub corpus is stored in the folder subcorpus_10000.

Assumptions:

For BM25 I used the formulation 11.32 from the textbook. I used ln for the calculation of idf. For the length of a document I assumed that the length is defined by the number of words it contains. To count the number of words it contains I used nltk.tokenize and removed all punctuations before counting the number of words. I removed the punctuations since for the word count because the index does not also not contain punctuations. With those assumptions the BM25 score is more precise. For the AND query I sorted the result by DocumentID. For the OR query I ordered the result by how many keywords the document contains and if two documents contain the same amount, the documents are sorted by documentID. For the query based on BM25 the whole ranked list of documents is returned.

Design Ideas:

For the query processor based on BM25 I decided to return the full list of ranked documentIDs. In my first implementation I used a heap to store the documentIDs and respective scores and only returned the top 10 ranked documents. My TA explained to me that for easier comparison the full list would be better. I commented out the code part which implements only returning the top 10 ranked documents since it is probably a better implementation in a non-academic setting.

I also implemented the selection of the queries as a console input. Therefore, the user does not have to change the code to use the query processors.

I separated the preprocessing of the text and tokens and the indexer, so that the time comparison between the SPIMI inspired indexer and the naïve indexer is more precise.

I also tested different values for the token limit (10000, 100000, 500000). With a higher amount of tokens in the sub corpus the time difference between the two indexers is even more obvious.

The different indexes that are created for the subcorpi are stored in files with the following names index_naive/spimiNUMBEROFTOKENS.txt.

For the spimi inspired indexer I did not use any blocks for which separate indices are created and later merged, because for our exercise the index of the full corpus can fit into the memory.

I also removed the punctuations from the input of the queries because the punctuation was also removed when constructing the index.

For the calculation of the idf I used ln(N/df) since that assumptions was also made in the textbook.

I tested different values for b and k1 for the BM25 formula. By looking at the formula I realised that k1=0 is the same as not considering tf at all. With testing I found out that having k1 between 1.2 and

1.5 seems to be the best choice. Since we are dealing with rather short news articles the k1 in that range seems to returning the best results.

For b I used 0.8 a larger b value seemed to penalize articles longer article. The reason behind that is that longer article can contain many different topics and keywords and are therefore more likely to be irrelevant even though they contain many of the keywords in the query. Since we are dealing with news articles, longer articles mostly touch on many different topics and not go into detail of one specific topic. Therefore a higher b value seemed to work better for me.

The query results for the test and sample queries are stored in txt files. Since many queries return long lists I did not want to copy the full list into the report and therefore stored them automatically in txt files. To create these files the user just has to type t in the console. Then all the test and sample queries are run and the results are saved.

What have I learned from the project:

```
C:\Users\Silas\PycharmProjects\COMP479_Project3\venv\Scripts\python.exe C:/Users/Silas/PycharmProjects/COMP479_Project3/main.py
The SPIMI-inspired indexer is 0.012311458587646484 seconds faster than the naive indexer (10000 tokens).
The SPIMI-inspired indexer is 0.15018415451049805 seconds faster than the naive indexer (100000 tokens).
The SPIMI-inspired indexer is 0.9922561645507812 seconds faster than the naive indexer (500000 tokens).
```

The time comparison between SPIMI and naïve indexer made me realise that especially for larger corpi the efficiency of an indexer is very important. When you think about web search engines the indexer needs to be as efficient as possible because every minor improvement has a high impact in that scale.

The different types of Queries I implemented helped me understand the advantages and disadvantages of the different types better. For example, AND queries can be helpful when searching for specific names. Other realisations are explained in the analysis of the test and sample queries.

Also, the implementation of the BM25 query processor helped me to understand the BM25 formula and the logic behind it fully. Especially trying out different values for k1 and b was interesting and showed what the different parameters do and how that can effect the ranking.

Test Queries:

The results can also be found in AND_sample_query.txt, OR_sample_query.txt, singlekeyword_sample_query.txt and bm25_sample_query.txt

A) **Single keyword query**

article

Uncompressed Index of Project 2: 41 documents contain the term 'article': ['925', '982', '1022', '1552', '1904', '2036', '2796', '4868', '5917', '6112', '6382', '6384', '7260', '8118', '8195', '8203', '8441', '8756', '9327', '10036', '10504', '10623', '11110', '11768', '11918', '12431', '12750', '12879', '13949', '15040', '15369', '16607', '16649', '17119', '17776', '17806', '18700', '18823', '18923', '19263', '21123']

New index with tf and idf:

41 documents contain the term 'article': ['925', '982', '1022', '1552', '1904', '2036', '2796', '4868', '5917', '6112', '6382', '6384', '7260', '8118', '8195', '8203', '8441', '8756', '9327', '10036', '10504',

'10623', '11110', '11768', '11918', '12431', '12750', '12879', '13949', '15040', '15369', '16607', '16649', '17119', '17776', '17806', '18700', '18823', '18923', '19263', '21123']

The results of the query on both indices are the same as expected.

I also tested the query article on the BM25 with interesting results:

Top 10 Documents according to BM25:

['4868', '2796', '11110', '6384', '19263', '18823', '8118', '8195', '7260', '8756']

All the documents in the top 10 are also contained in the result for the unranked retrieval since when the keyword is not in the document the documents get a score of 0. So, all the documents that contain the article at least once are in the top 10. According to the index document 2796 contains article 6 times and document 4868 contains article 3 times. That is the reason why these articles are so highly ranked. Document 2796 is much longer than article 4868 therefore 2796 is ranked lower even though it contains article double as often.

B) **BM 25**

average net profits

Enter the term/terms you want to search for. average net profits

Ranked List of documents according to BM25: ['3273', '17469', '15389', '5429', '15451', '17889', '10612', '6121', '6597', '19549', '5591', '15913', '1103', '16105', '4732', '17189', '15289', '6075', '2602', '21269', '17331', '696', '971', '17921', '8757', '4074', '8449', '5787', '5647', '19937', '8475', '6376', '1260', '9232', '15841', '11898', '15427', '21260', '10715', '8595', '11827', '16367', '16893', '20375', '21145', '317', '6363', '6277', '10840', '17107', '19068', '19469', '9707', '12682',..]

Scores:

[12.673112094861484, 10.826561782175308, 10.635919089732434, 9.619861537156662, 9.540794047937773, 9.198717873547004, 9.167224934366676, 9.141701830984125, 8.698926557036483, 8.692017547524546, ..

```
<TITLE>DUTCH RETAILER AHOLD SEES UNCHANGED 1987 PROFIT</TITLE>
<DATELINE>    ZAANDAM, Netherlands, March 9 - </DATELINE><BODY>Dutch food retailer Ahold
NV &lt;AHLN.AS> expects unchanged profits in 1987 but said it will
take advantage of the lower dollar to expand further in the
U.S.
    Turnover will grow but net profits are expected to remain
around the 1986 level of 132.4 mln guilders due to higher Dutch
taxes and a three-year expansion plan, Ahold Chairman Albert
Heijn told a news conference.
    The profit forecast allows for a dollar rate around current
levels of just over three guilders.
 Turnover and net profit in 1986 were hit by the dollar,
which fell to an average 2.46 guilders from 3.35 in 1985.
 Reuter
```

The article that has the highest score deals with the profits of a Dutch food retailer. The article short and contains all the keywords and even contains some keywords more than once.

C) AND QUERY

San Francisco

43 documents contain the terms ['San', 'Francisco'] : ['67', '215', '367', '805', '863', '1402', '2444', '2740', '3038', '3521', '3561', '3992', '4622', '5719', '5730', '6278', '6766', '8352', '8886', '10460', '10998', '11042', '11863', '12066', '12072', '12667', '13145', '13877', '14938', '15240', '15274', '15443', '15484', '15946', '17676', '17680', '17974', '18075', '19828', '19890', '20236', '20897', '21292']

```
<DATELINE>    Kansas City, March 11 - </DATELINE><BODY>The National Weather Service said
heavy afternoon and evening rainfall caused a threat of
flooding over portions of Texas. There were reports of more
than two inches of rain falling across Terrall and Val Verde
counties during the afternoon, which caused street and highway
flooding.
    A coastal flood watch was also posted along the outer banks
of North Carolina and warnings of gale force winds have been
posted along the central Atlantic Coast.
    Cold temperatures and rain mixed with snow caused the
advisories for livestock which were posted over northwest Texas
through the morning.
    Rain continued over central Texas into the morning hours.
Rain also extended over the Oregon coast, near San Francisco
Bay, California and over northwest Montana.
    Snow was scattered over northwest Texas, as well as from
northeast Montana across North Dakota into western Minnesota.
    There were no reports of measurable snowfall during the six
hours ending at one am est.
```

The query processor works as expected. One of the articles is a weather report. It contains many states, regions and states. This example demonstrates that the unranked AND query is not the most suited for retrieving the most relevant articles in some cases. Even though the query is not that specific the user was probably searching for articles that are mostly focusing on San Francisco and not articles that talk about many states and regions in the US.

D) OR QUERY

Czech Republic Luxembourg

300 documents contain the terms ['Czech', 'Republic', 'Luxembourg']: ['12486', '21533', '291', '301', '329', '414', '421', '748', '905', '923', '928', '931', '1005', '1056', '1086', '1089', '1136', '1141', '1163', '1356', '1424', '1474', '1475', '1519', '1547', '1559', '1579', '1922', '1928', '1950', '1977', '1978', '1992', '2077', '2088', '2143', '2158', '2170', '2200', '2245', '2257', '2272', '2366', '2423', '2491', '2512', '2532', '2610', '2646', '2655', '2997', '3000', '3136', '3488', '3503', '3581', '3606', '3641', '3644', '3648', '3686', '3718', '3938', '4071', '4073', '4095', '4116', '4222', '4354', '4488', ..]
The first document contains 2 of the terms, the second document also contains 2 of the terms, the rest of the documents only contain one of the keywords.

I chose this example to demonstrate how you can find specific articles with the ranked OR query. The first two articles are the only articles that contain two of the terms in the query. SO they are probably the most relevant even for a more advanced ranking system. But this query also shows

that the major problem of the OR query is the fact that most of the times a high amount of documents are retrieved.

Information needs

I did not copy the full list of results for every query because the lists were very long for most of the queries.

Full list of results can be found in bm25_test_0/1/2.txt, AND_test_0/1/2.txt and OR_test_0/1/2.txt.

BM 25

Information need: Democrats' welfare and healthcare reform policies

Ranked List of documents according to BM25: ['20449', '7433', '18722', '6940', '4006', '8072', '5868', '9248', '9096', '11204', '7219', '3619', '7375', '3899', '19134', '17940', '10230', '4268', '18161', '16092', '18683', '2883', '2891', '18731', '12806', '2417', '18469', '12552', '12750', '8310', '7401', '16171', '10212', '4882', '8139', '8307', '951', '9704', '7019', '20023'..]

Searching for Democrats instead of Democrats‘

Ranked List of documents according to BM25: ['20449', '7433', '18722', '21577', '6940', '18878', '14976', '4006', '8072', '5868', '18731', '9248', '18271', '9096', '1999', '11204', '7219', '15661', '219', '3619', '7375', '20805', '12326', '7467', '3899', '19134', '9688', '17940', '18357', '9347', '13257', '10230', '4268', '13671', '16048', '19660', '18161', '16092', '7806', '18683', '2883', '2891', '12806', '1908', '2132', '2417', '20890', …]

I noticed that there is a difference when running the query: with Democrats rather than Democrats’. The reason behind that is that Democrats’ is not stored as a term in the index because even when Democrats’ is mentioned in an article is is split into the tokens Democrats and ‘. Since Punctuations are removed the word Democrats in the query produces somewhat different results. But it is surprising that the results do not differ much. The top three documents are the same and only minor differences in the ranking can be seen in the documents after the top three.

Below are the first two ranked articles both of them are not relevant to the query. But because both articles are so short and both contain the word ‘healthcare’ which is probably very rare they are ranked so high. As the AND query seen below shows there is no article which contains every term in the query.

```
<DATELINE>    MOUNT LAUREL, N.J., Oct 20 - </DATELINE><BODY>Biosonics Inc said Jeffrey S.
Wigand has resigned as president, chief operating officer and
director of the company.
    The company said Wigand will announce his affiliation with
another healthcare company as soon as negotiations are
completed.
 Reuter
&#3;</BODY></TEXT>
```

```
<DATELINE>    FLEMINGTON, N.J., March 19 - </DATELINE><BODY>FCS Laboratories said its
investment banker, Butcher and Singer Inc, received a
preliminary merger proposal from another company in the
healthcare field.
    FCS said that if various aspects of the proposal are better
defined, its board may consider it. FCS said merger
negotiations with this other company have been continuing since
late August.
 Reuter
```

Information need: Drug company bankruptcies

Ranked List of documents according to BM25: ['16771', '4050', '8209', '16994', '9542', '13764', '1805', '6089', '21239', '18716', '11585', '3577', '3328', '17520', '16219', '19640', '7125', '21348', '10026', '14401', '17604', '19720', '6544', '11434', '20953', '9262', '12242', '3176', '11313', '1819', '3106', '1391', '2868', '11525', '14165', '11553', '21069', '4435', '9546', '21051', '3776', '3297', '3756',..]

Searching for drug instead for 'Drug'

Ranked List of documents according to BM25: ['16771', '4050', '8384', '19640', '9542', '8209', '1998', '9342', '3108', '16413', '192', '2356', '19437', '21348', '3739', '17680', '20953', '6378', '4408', '457', '2523', '14359', '4871', '15088', '730', '21051', '7125', '16829', '7940', '18808', '16259', '9546', '12903', '2093', '826', '5319', '11613', '13068', '19292', '3091', '7753', '6031', '5486', '9633', '20892', '19691', …]

Both queries return 16771 as the highest ranked document. The article is short and deals with bankruptcies (as seen below). But now with drug company bankruptcies. The other articles in the top three also only deal with corporate bankruptcies and not specifically drug companies' bankruptcies. There is either no article that specifically only deals with drug company bankruptcies or the ranking ranks these articles not as high as the sort articles that talk about bankruptcies in general.

```
<TITLE>FRENCH CORPORATE BANKRUPTCIES RISE IN MARCH</TITLE>
<DATELINE>    PARIS, April 17 - </DATELINE><BODY>French corporate bankruptcies rose to a
seasonally-adjusted 2,857 last month from 2,631 in February and
2,572 in March 1986, the National Statistics Institute (INSEE)
said.
    The rise has been progressive since the end of last year,
with bankruptcies totalling 2,367 in January and 2,195 in
December 1986.
    The cumulative total for the first quarter of this year was
7,855 bankruptcies, four pct up on 7,560 in the first quarter
of 1986.
```

Information need: George Bush

Ranked List of documents according to BM25: ['8593', '20891', '4008', '16780', '20719', '16824', '3560', '2711', '8500', '7525', '20860', '2766', '4853', '10400', '6564', '2796', '5459', '2733', '15284', '10682', '16115', '5334', '255', '21393', '7469', '16229', '17150', '17647', '286', '871', '15478', '8554', '12009', '2356', '6423', '6065', '16318', '43', '1667', '8053', '18160', '13507', '16730', '4737', '8748', ..]

For these result is was very interesting to compare it to the results of the AND and the OR query. It showes that BM25 also takes the length of the article into account and even ranks some long articles that contain both George and Bush lower than short articles that only contain one keyword. I think at least with my values for b and k1 an AND query or the ranked OR query seems to be more suitable when searching for a specific name.

AND Query

Information need: Democrats' welfare and healthcare reform policies

No document contains the terms ['Democrats’', 'welfare', 'and', 'healthcare', 'reform', 'policies'].

Searching for Democrats instead of Democrats’:

No document contains the terms ['Democrats', 'welfare', 'and', 'healthcare', 'reform', 'policies'].

These results show that for very specific queries with many keywords using an AND query is not suitable. Since especially with a smaller corpus many of these queries do not produce any results.

Information need: Drug company bankruptcies

No document contains the terms ['Drug', 'company', 'bankruptcies'].

Searching for drug instead of Drug:

No document contains the terms ['drug', 'company', 'bankruptcies'].

These results show the same as said before and also explain the fact that the top documents for BM25 do not specifically deal with drug company bankruptcies. Maybe there is no article that deals specifically with this information need.

Information need: George Bush

13 documents contain the terms ['George', 'Bush']: ['854', '965', '2796', '3560', '4008', '5405', '7525', '8500', '8593', '16780', '20719', '20860', '20891']

These results show that when searching for names an AND query can be very suitable. Also if the user searches for a specific term or concept that needs to be named in the article the AND query is the best choice. It is also interesting that the result differ to the results of the BM25 query. Some articles like 854 are not even in the top 25 of the ranked result of BM25.

OR Query:

Information need: Democrats' welfare and healthcare reform policies

14176 documents contain at least one of the terms ['Democrats’', 'welfare', 'and', 'healthcare', 'reform', 'policies']: ['672', '862', '1895', '2132', '2417', '4006', '5868', '6606', '6940', '7219', '7375', '7493', '7892', '8072', '8310', '9055', '10230', '10355', '11204', '12552', '12598', '12750', '12806', '12879', '14874', '15382', '16092', '16168', '16226', '16544', '17070', '17185', '17402', '17940', '20462', ..]

Searching for Democrats instead of Democrats’:

14181 documents contain at least one of the terms ['Democrats', 'welfare', 'and', 'healthcare', 'reform', 'policies']: ['1895', '2132', '28', '672', '862', '1999', '2417', '4006', '5868', '6606', '6940', '7219', '7375', '7493', '7892', '8072', '8310', '9055', '9774', '10230', '10355', '11204', '12552', '12598', '12750', '12806', '12879', '14874', '15369', '15382', '16092', '16168', '16226', '16544', '17070', '17185', '17402', '17915', '17940', '18271', '18328', '18357', '18731', '18878', '19660', '20462', '20805', '21508', '18', '32', '54', '55', '61', '179', '181', '209', '219', '225', '226', '234', '236', '237', '238', '263', '308', '335', ..]

As mentioned already the or query produces a long list of results for multiple keyword queries. For this query and probably for most rather specific information needs the OR query does not produce the best results. The reason for that is mostly that it only takes the number of keywords contained into account and not more parameters that we know play an important role for relevance like the BM25.

Information need: Drug company bankruptcies

5311 documents contain at least one of the terms ['Drug', 'company', 'bankruptcies']: ['192', '696', '730', '1391', '1805', '1819', '2271', '2868', '3106', '3176', '3297', '3328', '3577', '3589', '3682', '4050', '4871', '5319', '5711', '6089', '6544', '6608', '6996', '7163', '7260', '7753', '7940', '8384', '9262', '9331', '9342', '9542', '9579', '9585', ..

Searching for drug instead of Drug

5324 documents contain at least one of the terms ['drug', 'company', 'bankruptcies']: ['192', '375', '457', '696', '730', '902', '1269', '1391', '1998', '2036', '2127', '2356', '2523', '2636', '3091', '3108', '3205', '3739', '4050', '4408', '4485', '4871', '5319', …

These results show that as mentioned above the OR query returns a high number of documents for a multiple keyword query. Still the article 192 is ranked first even though it deals with a lawsuit against a drug company and not bankruptcies.

Information need: George Bush

140 documents contain at least one of the terms ['George', 'Bush']: ['854', '965', '2796', '3560', '4008', '5405', '7525', '8500', '8593', '16780', '20719', '20860', '20891', '28', '43', '178', '255', '286', '342', '871', '925', '1022', '1502', '1667', '1729', '2356', '2711', '2733', '2766', '2802', '3014', '3366', '3390', '3563', '3938', '4098', '4737', '4764', '4853', '5157', '5334', '5459', '5574', '5631', '5711', '6062', '6065', '6296', '6423', '6564', '6656', '6744', '6882', '6896', '6940', '6989', '7186', '7326', '7469', '7471', '7765', '8009', '8053', '8252', '8554', '8748', '8765', '9002', '9150', '9247', '9342', '9755', '10400', '10682', '10725', '11380', '11624', '12002', '12009', '12209', '12261', '12460', '12605', '12709', '12720', '12806', ..]

The results are the same as for the AND query for the first few documents since these are the documents that contain both keywords. After that documents that contain only George or Bush are listed. It is interesting that for BM25 the document 854 is not in the top 30 even though it contains both keywords. It is probably because the article is very long. But this result shows that when searching for name a ranked OR query could be suitable.

I certify that this submission is my original work and meets the Faculty's Expectations of Originality.

Signature ID: 40256077 Date 2022-11-07