**IR Assignment report**

ID: 2017HT12447

Name: Manish Singh

**How the algorithm works?**

The algorithm I have written generates positional indexes, uses stop words and also implements stemming/Lemmatization using porter algorithm principle.

The algorithm works on the following steps:

**Step#1**: The algorithm scans for the stop word from a file called “stop.txt”

**Step#2**: Once the stop words are scanned, it scans the documents given in the dataset one by one.

**Step#3**: For each document, it tokenizes it and then removes all the tokens which are mentioned in the stop list “stop.txt”

**Step#4**: All terms are camel cased i.e. converted into lowercase.

**Step#5**: The resultant tokens are then processed for stemming/Lemmatization using porter algorithm principle in a function called as **lemmatization.** It takes the word and returns it’s lemma.

**Step#6:** All the tokens are entered in a dictionary and it is ensured that the dictionary does not contain duplicate node.

**Step#7**: Once we have the dictionary in place, each document is scanned again to create a positional index. A sample index looks like:

**footer,4:{ Doc1.docx,1: {361}; Doc2.docx,1: {572}; Doc5.docx,1: {149}; Doc6.docx,1: {336}; }**

The first term is the index, second is the term frequency. It is then followed by a list which specifies Document name, term count, term position in that document.

The complete index is printed in a file called “**index.txt**”

**Step#8**: Once the index is created, a prompt is displayed which asks the user for a query. The code supports the following formats:

* <term>
* not <term>
* <term> and <term>
* <term> or <term>
* not <term> and <term>
* <term> and not <term>
* not <term> and not <term>
* not <term> or <term>
* <term> or not <term>
* not <term> or not <term>

Eg:

* across
* not across
* across and friend
* across and not friend
* not across and not friend
* across or not friend
* not across or not friend
* not across or friend

**How to use the executable?**

The assignment is written in Java 1.8 update 172, so jre 1.8 is requires as the jar is build on that. The assignment used Apache POI for reading .docx files. The same has been embedded in the standalone executable jar so no need to download it to run the executable.

IMP: As taxilla is not allowing more than 10MB and my jar is 13.5 MB (winzip and 7z are not able to compress it), I am uploading my jar to my site. You can download the same at:http://download.manishsingh.in/ir\_assign\_with\_lemma.jar

Alternatively, I have uploaded a split zip file using winzip under the folder: Ir\_assignment\_withJar-Splitzips (in taxila). You can download it and merge them back using winzip (right click on the main.zip file and click extract here.. or similar. It will do the extraction). You would need winzip for this. That zip has the jar file.

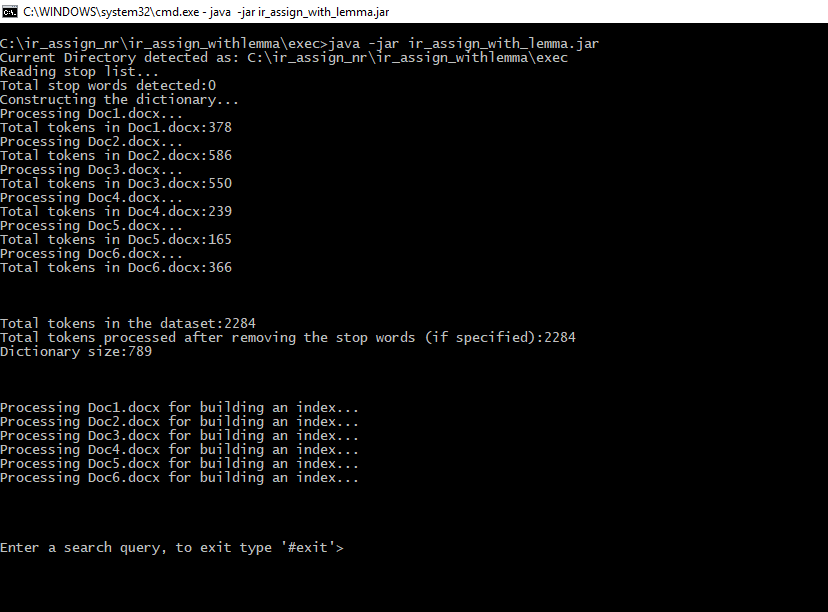
**I have also emailed the jar to mam as it went via gmail.**

In the zip, go to executable folder and type the following:

**java -jar ir\_assign\_with\_lemma.jar**

The tool will start and create an index inside index\_output folder. It will display some messages like token count, dictionary count etc. Allow it 10-30 seconds to complete. The tool will take some time at the message “Processing Doc1.docx”. Once done you will get a prompt like

**Enter a search query, to exit type '#exit'>**



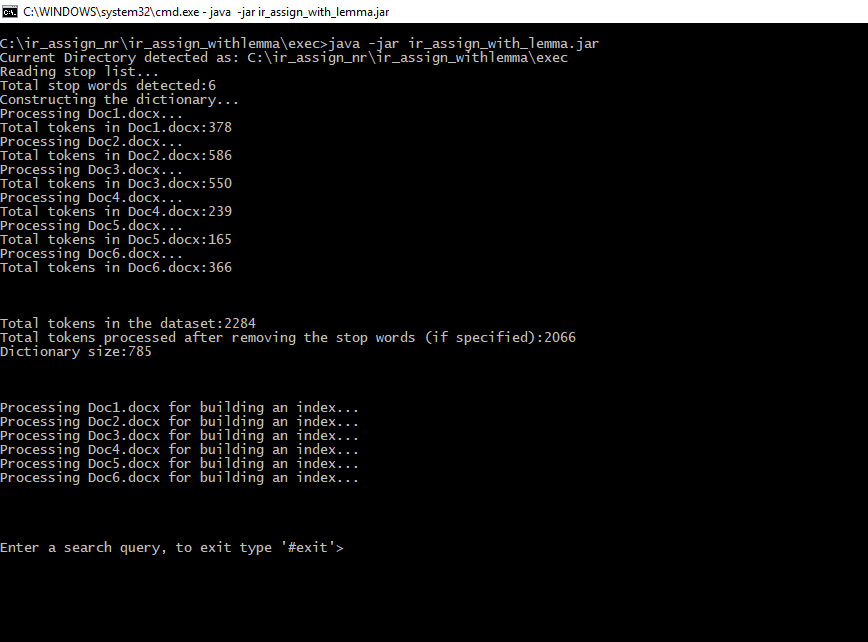
The tool looks in the following folder:

1. **Irdataset** – To scan the documents
2. **stop\_words**\stop.txt – To scan the stop words
3. **index\_output** – To write the index output.

These folder are already created indie executable folder and are required for the proper run of the tool.

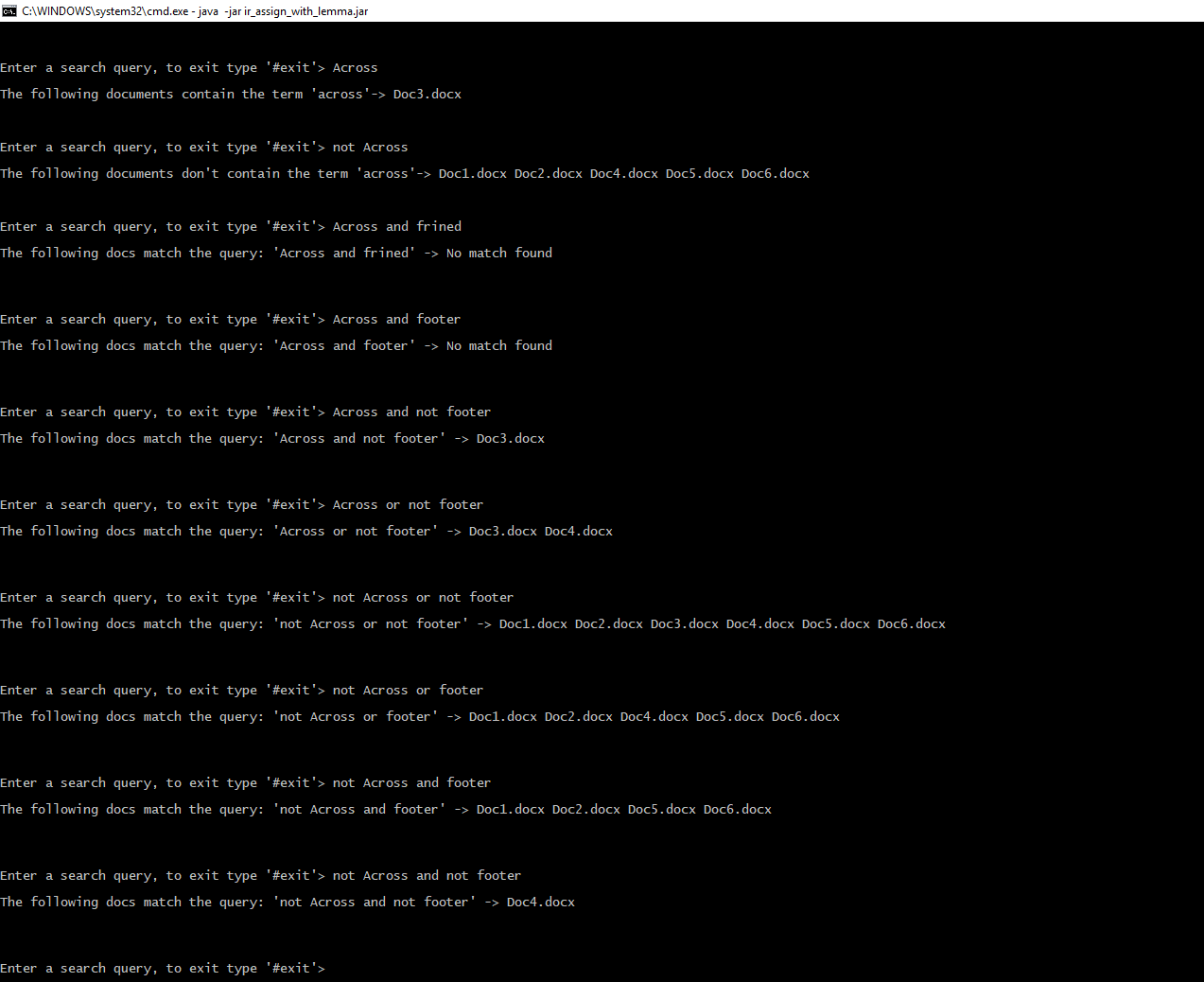
Screenshot with some stop words: and,or,is,the,a,an

**Note**: The tool expects stop words comma separated



Let’s see some queries

1. Across
2. not Across
3. Across and frined
4. Across and footer
5. Across and not footer
6. Across or not footer
7. not Across or not footer
8. not Across or footer
9. not Across and footer
10. not Across and not footer



See Conjunctive \_quries\_op .txt for easy reading

**What is observed?**

**For the submitted index output**

|  |  |  |
| --- | --- | --- |
| **Index file name** | **Dictionary count** | **Stop words used** |
| index\_noStopWords.txt | 789 | No |
| index\_withStopWords.txt | 784 | Yes |

**Stope words are in:** .\executable\stop\_words\stop.txt

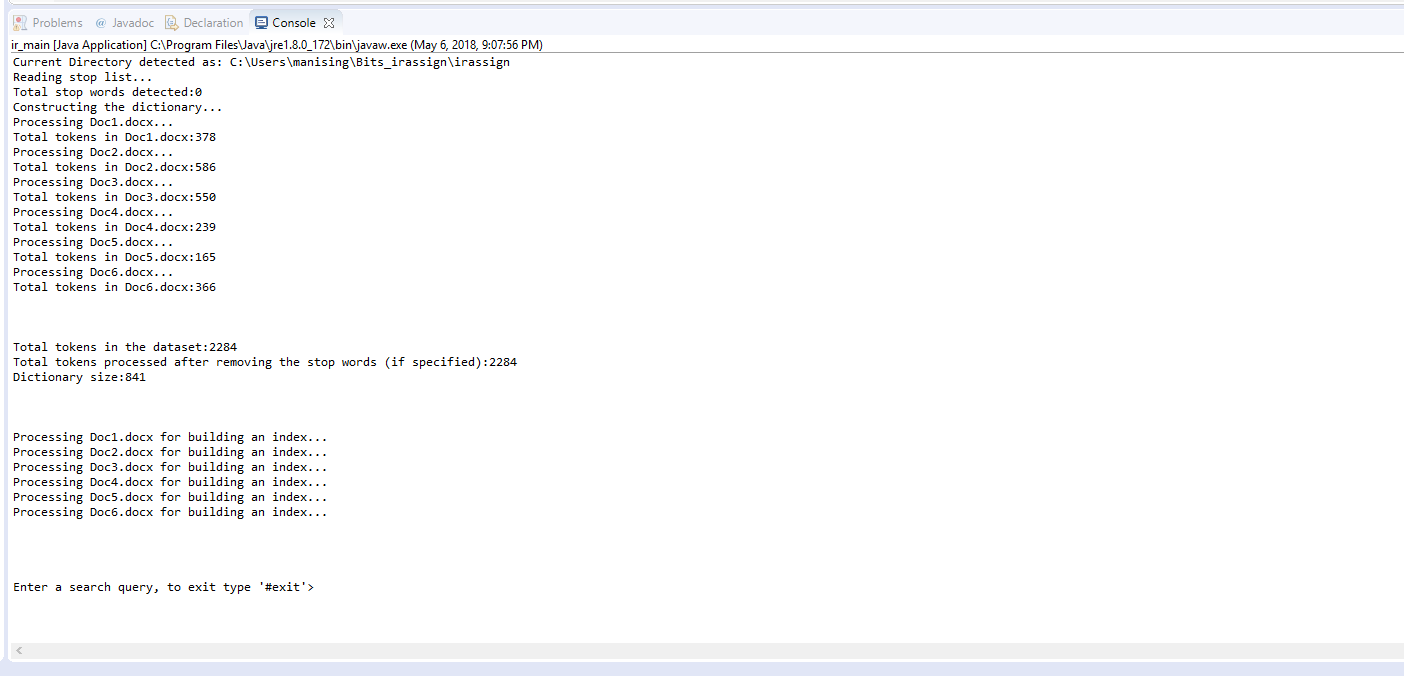
**Additional observation:**

Without stemming/Lemmatization and stop words, the dictionary size is **841**

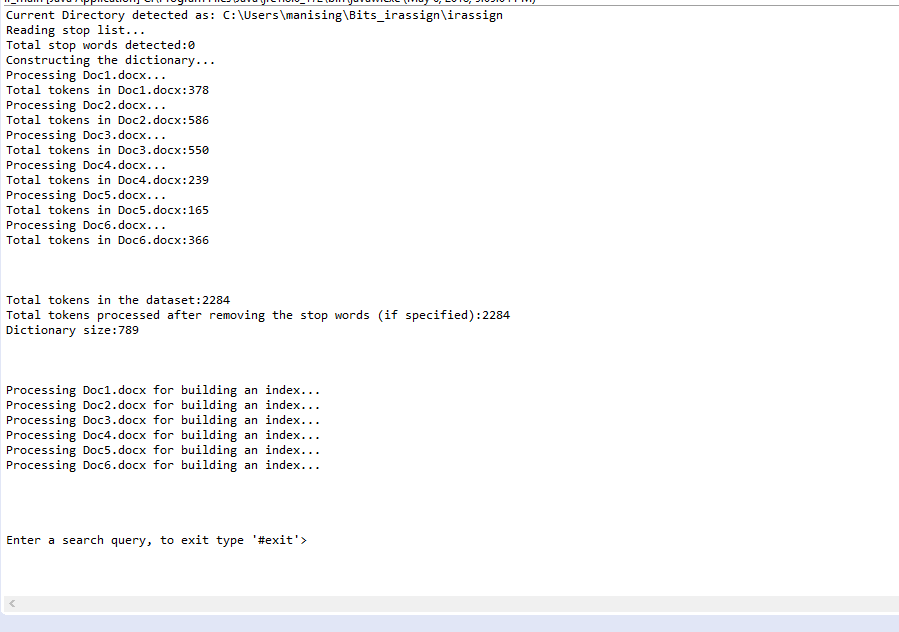
With stemming/Lemmatization and without stop words, the dictionary size is **789**

So, stemming/Lemmatization is reducing our dictionary size.

Without lemma



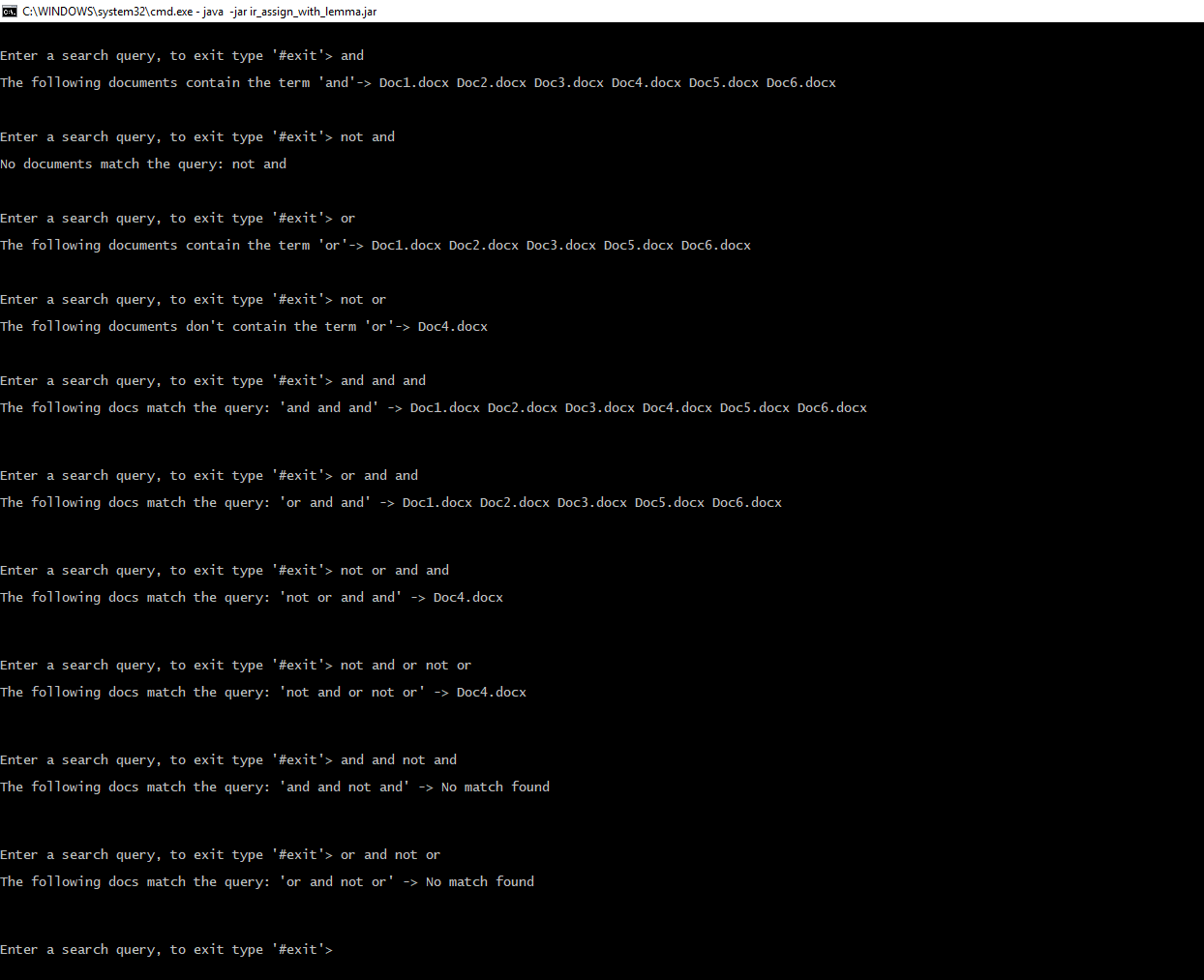
With lemma



**Note**:

The tools also supports queries like (to use them, remove and,or from the stop word list located at .\executable\stop\_words\stop.txt and re-run the tool so that an index is regenerated):

* And and and
* Or or or
* And or and
* Or and or



Output of the screen: Conjunctive \_queries\_op1.txt