**Task 1: A Description of the process involved in creating Inverted index**

An inverted index is useful because it is a method of breaking down documents that consist of all unique terms and covers an entire details of a document. It is useful in IR because it forms the core functionality of IR process and it is to be considered for every efficient IR. it is used to retrieve information from a large range of document fast. The basic component of an inverted index are dictionary and posting.

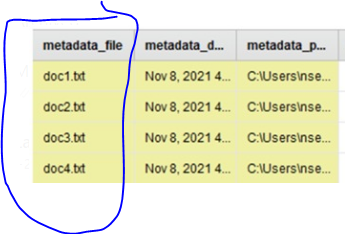
The key task needed in other to create an inverted index are;

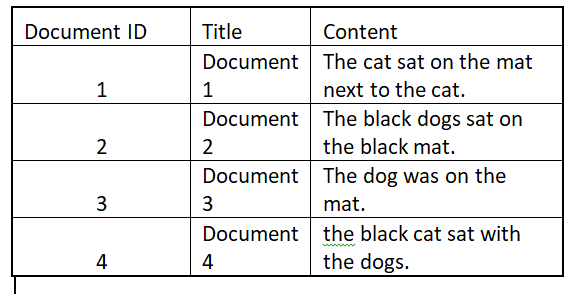
* Collecting all documents (Document indexing)
* Tokenizing
* Normalized Tokens/Preprocessing
* Stop Words
* stemming

The process of using Rapid miner for the IR Documents provided.

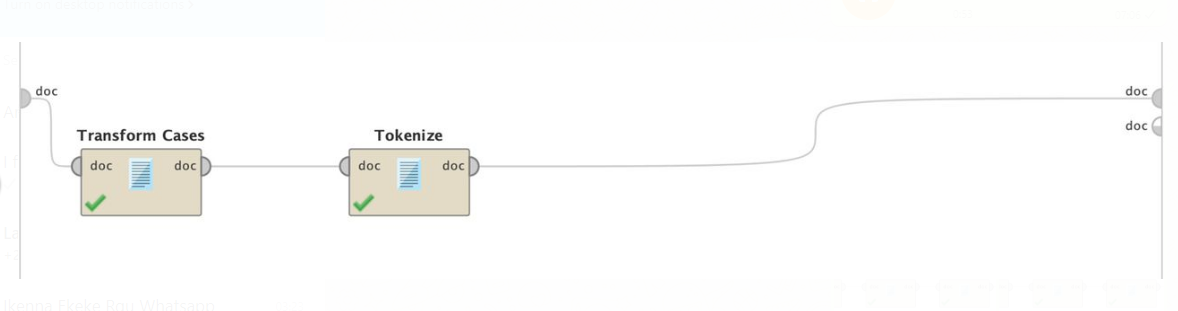
Document indexing is Collecting all document is the state of relating a content of a document to the file. Open the Rapidminer and import the file from the location and import the given process (Import process) The document will run to make the document. Also representing it by serial numbers called the ID thereby associating the information with a file so it can be easily located or retrieved later. The below images show the process in rapid miner and the details of the documents we are to see.

The circled path shows the number of documents using rapid miner IR process



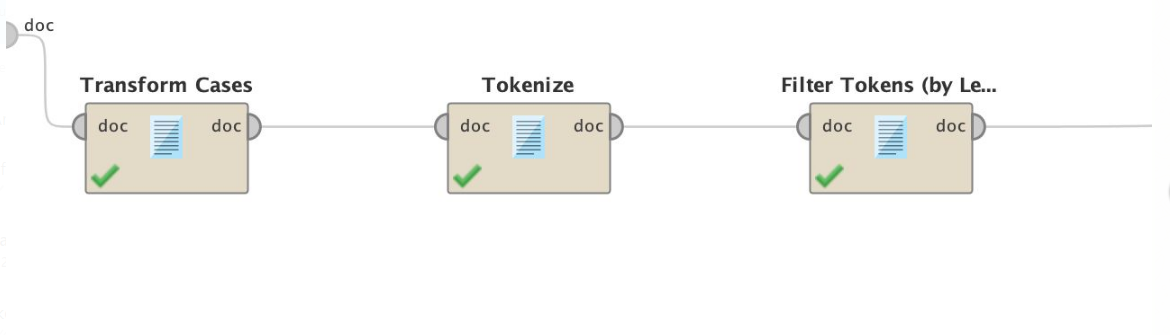


**Tokenizing**; This is the act of turning each document into a list of token/ breaking down into individual components thereby removing the special characters as in the IR engine such as (.)and also producing terms from each documents





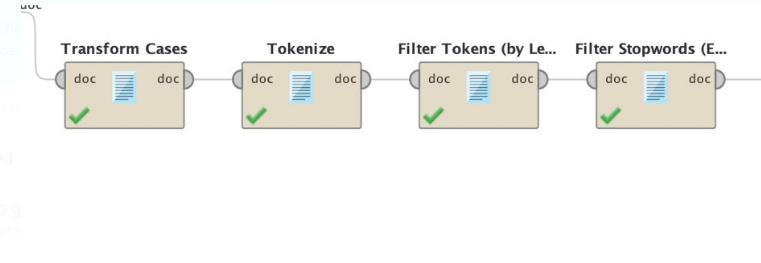
Afterward we do a list of preprocessing and produce normalized word and then sort by terms and ID as shown below;



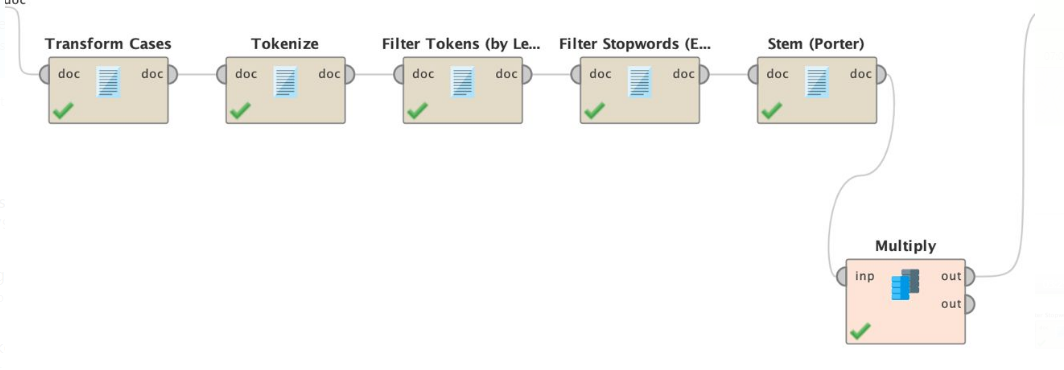


**Stop words removal** : The are words that have little value in helping select documents matching/ the words that have high use in the query will be excluded such **eg Document 1; “The cat sat on the mat next to the cat**” with “string stops” where the words **on,the,to e.t.c** and the machine will not return the words as listed because they have been recognized as **stopwords.**





**Stemming**: This is used to remove prefixes and suffixes in words for example **dog** and **dogs** in document 2,3 and 4 or This is a process of reducing words to its word stem. However, the words are stemmed to create more space for memory. If stemming is **true** ? the search word **dog** will retrieve the **all documents that contain dog** and if we set otherwise for stemming (**false**) the search for the word dog will retrieve only the document with dog and not the documents that contain the plural form also if alternatively, we search for **dogs** the term released will only be the case for document 2 and 4 respectively.



**Inverted Index file (The result of dictionary and posting) below;** This is the result of the documents and the document frequency when stemming is true in inverted file with statistics when building index



**Task 2A: Comparison of inverted document with Stemming and not stemming using IRDocuments**

* The storage space and retrieval efficiency of an inverted file when stemming is used creates spaces in the memory, it also reduces the total number of index terms when stemming is used compared to when stemming is not used.
* Stemming is fast and able to handle irregular plurals, a fewer terms is required for index building when stemming is used as otherwise when stemming is not used.
* The number of posting list increases when stemming is turned off but it is reduced when stemming is on.
* The terms in index is usually less when stemming is used because the terms are considered when stemming is used however when stemming is not used the index term are usually more because every term is represented exactly the way they appear and no predictions is assumed.
* Stemming removes the affixes from the word in other to attempt to find the root while not stemming repeats every word with its own attributes in every instance(actual word).
* With stemming on sample query of a particular word there will be a number of document returned which will produce either an extended version of the likely words **like how google search** works meanwhile without stemming will produce just the exact words without further predictions.
* Non stemming in the instance will be best to use when the meaning of every word is important for analysis meanwhile stemming is used when words are not important in analysis for example hence the reason I stemmed my document above.

For example using the image below the inverted index of IRdocuments that were stemmed



The statistics of the inverted file when **stemmed** are calculated below;



Meanwhile when documents are **not stemmed** the inverted files are as represented below;



The statistics of the inverted file when **not stemmed** are calculated below;



**Task 2A: Comparison of inverted document with Stemming and not stemming using Abstract;**

The statistics with stemming is as shown below;

The total number of document is 999

The total index terms from document 86865

The Number of posting list is 4725

The total number of Postings is 55674

Average no of posting per posting list 11.7828571

**Meanwhile, the statistics without stemming**

The total number of document is 999

The total index terms from document 86882

The Number of posting list is 8442

The total number of Postings is 61000

Average no of posting per posting list 7.22577588

**Task 2B: Compare retrieval effectiveness for the specified queries.**

According to the table below the values for retrieval of the IR process with stemming and without stemming for the specified queries.



It also explains the precision value when P@5 and P@10 for stemming and non stemming on the learning to rank algorithms.

The image implies that when **stemming is on** the total number of document that was relevant document retrieved is 3 when p@5 is 0.6 and when P@10 the total number of retrievd relavant document is 0.5.

Otherwise when **stemming was off** when P@5 is 0.4 and when p@10 it is 0.5.

And for the automated query learning it also implies that when stemming is on P@5 is 0.6 and P@10 is 0.6

Alternatively when stemming is off P@5 is 0.4 and P@10 is 0.5



Therefore with the explanation above shows that precision is the number of relevant document retrieved(the useful items or terms that are useful to a user or that matches the users needs) in a query and the higher the relevant the more constant precision value will be. Recall simply measured the extent at which a system that is processing the query above is able to retrieve relevant document that we want to use

Stemming is a recall enhancing device because applying stemming is specially relevant for document retrieval it may also allow to improve recall without a loss in precision when document re being queried.

In summary, when stemming is on the total number of document retrieved is always more for every retrieval because retrieval is always done for stem of a particular term and all the relatively extended versions will be derived as well but without stem/ non stem retrieval will be done based on the actual term in the query. Which makes it less effective in IR.

**Subtask 3A: Calculate the precision and recall for the specified queries with the 3 weighting functions.**

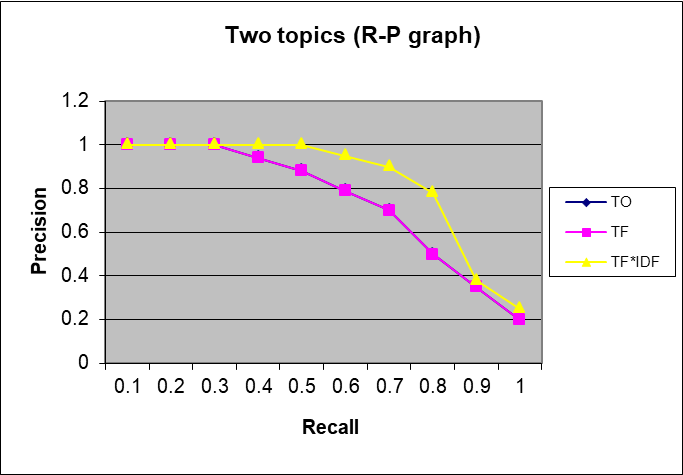
Below is the snapshot of precision and Recall at each relevant document rank for the 3 functions TO, TF, TF\*IDF for the 2 Queries

The total number of relevant document for Q1 is 6 and for Q2 is 4





The 2 Topics averaged R-P graph for each average weighting as computed is shown below.



**Subtask 3B: Discuss Topic Retrieval and Weighting Function Retrieval**

Using the **performances of the individual queries** and the impact of the weighting function I have for probabilistic models for socials query.

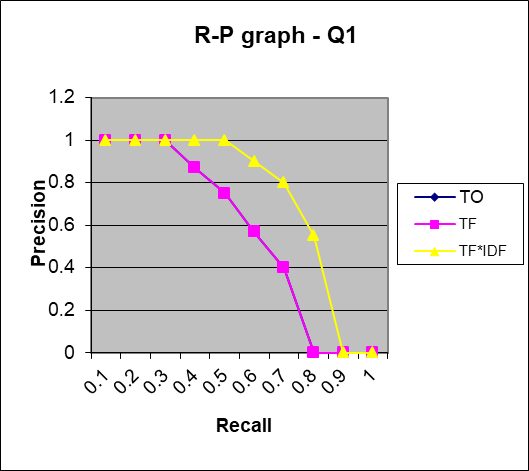
In the **TO (Term Occurrences)for probabilistic models for social**, I have the total of 4 relevant document and 6 irrelevant document. When precision value was calculated it showed that it was only constant at the first 2 instances which were relevant then the moment the irrelevant document were retrieved the value of precision in the query further reduced from 0.75 to 0.4. and the final recall value is 0.67. Then I calculated recall based on the number of relevance document that was retrieved across the query which was 6.

In the **TF(Term Frequency) for probabilistic models for social**s I have a total of 4 relevant document and 6 irrelevant document from the total number of documents retrieved, the first 2 retrieved document were relevant which caused precision to be high at that point but the 3rd was irrelevant which caused the precision for the 4th was affected and dipped to 0.75 but also the sequence changed and as more number of irrelevant document were further retrieved.

If we noticed the number of the document retrieved in TO are also in sequence as the same number of documents retrieved in TF, the recall as calculated in this case also had 4, they both have the maximum value in recall as 0.67. So therefore the number of relevant document, precision and recall for TO and TF are the same.

In **TF\*IDF for probabilistic models for socials** , I have 5 relevant and 5 irrelevant document from the total number of documents retrieved unlike the TO and TF I have a 3 constant value of relevant document that dropped after the 3rd retrieved document causing a drop in the precision for the 4th relevant retrieved document. However there is an equal instance of relevance and non relevant document in this case but with different instances. The recall and precision final values of these were 0.8 and 0.55 respectively.

To conclude the case of Query 1 (Q1) the TF\*IDF weighting function is just slightly more effective because it has equal weight for more words which is slightly higher than TO and TF in the query, the TF\*IDF application produced more but equal rate of relevance document and a slightly high value of precision compared to other weighting functions. Please see below graph for graphical details.

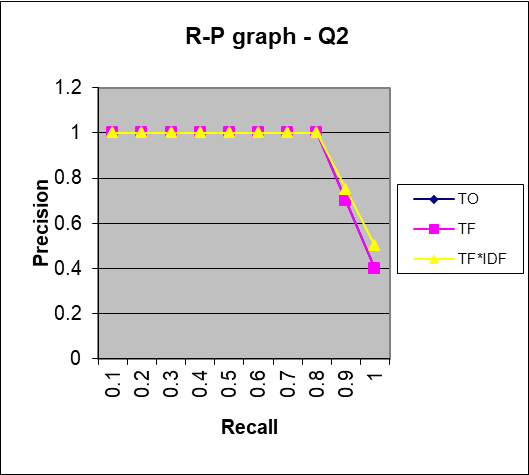


In the **TO(Term Occurences) for the Edit Distance Metrics** when weighting function was applied I have 4 numbers of relevant document and 6 numbers of irrelevant document in the total of 10 documents. where I have the first 3 retrieved document were relevant making the precision at maximum but as the retrieval continued the higher the number of the irrelevant document which brought about a reduced the precision to a value of 0.4 and recall at 1.

In the **TF(Term Frequency) for the Edit Distance Metrics** when weighting function was applied we also got the same number of relevant document retrieved R= 4 and N=6 in the total of the 10 documents. Similarly where the first 3 retrieved document were relevant the precision was also at maximum and reduced as retrieval continued and the irrelevant document were more populated which made precision drop to a value of 0.4 and recall at 1.

In the **TF\*IDF for the Edit Distance Metrics**  the weighting scheme has 4 relevant document and 6 irrelevant document in the list of 10 retrieved document. The first 3 retrieved document were relevant which led to high constant precision figures, then the recall values were calculated based on the total number of relevant document across the scheme which resulted to 4. However the only difference is the position of the 4th relevant retrieved value after the number of the 3 retrieved relevant document. The precision is closed at 0.5 and the recall is 1.

From the graphical representation below of Q2 it shows that there is a close call between this weighing function as they all produce 4 relevant document and the same number of irrelevant document but the precision of TF\*IDF position at recall 1 was just a point away from TO and TF relationship. This was based on when each of the weighting term was closed on recall and precision.



Comparing the impact of the 3 weighting scheme IDF TF TF\*IDF by the explanation above.

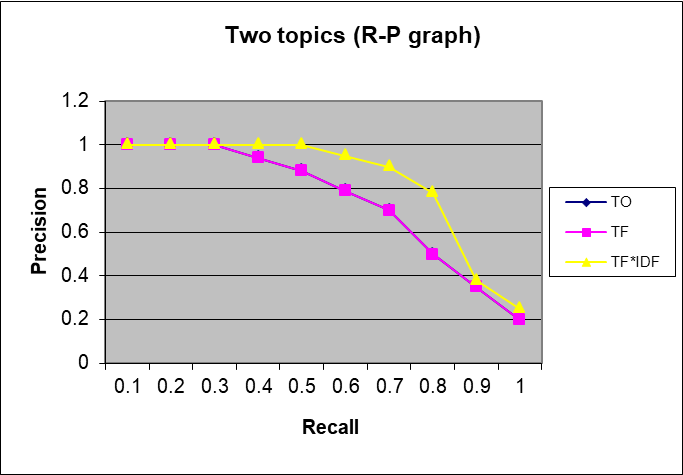


Similarly the highest recall value for each document across the queries equals 1.

I then have the average precision value for calculated for the 3 weighting schemes as per different queries. The image below shows the calculated average for the weighting scheme illustrating that there is almost an equal percentage of relevance in the TF\*IDF and higher relevance between the other weighting function.



However, also a graph that shows the average relationship between the two queries and the level of precision and recall for each query as represented showing to support my explanation. TO and TF are at equal average across the queries.



Thank you.