**Microblog Track**

Please see the problem statement here: <https://sites.google.com/site/microblogtrack/2012-guidelines>

You will need WordNet installed in the system you are running the code in. You would also need to update the path where WordNet is installed in the code.

Method:

I tried to overcome the “shortness” of the tweets by expanding the document using Wordnet. I built two indexes, the first was built on tweets using a lucene analyzer while the second was based on synsets obtained from the tweet words. I preprocessed the documents by removing retweets and tweets in languages other than english. I tried removing these documents in the indexing phase but the documents were a lot in number and it became a very long CPU intensive job, so i put these conditions in the searcher phase of the retrieval.  While building the second index i did put a check on language before obtaining synsets so that it would cut back on my time when i search for synsets in the WordNet dictionary. I removed duplicate words from the synsets because i thought they would influence the scoring in a very big way.  For the searcher phase i manipulated scores from both the indexes. For the AdHoc process i first obtained the first N results based on the queries in the topic file and a filter on the range of tweets valid. For each result i got i search for the same query in the second index with the document i obtained in the original result set as a filter value. For example if i get a result for java in the first index (which is index on the tweets alone) with a tweet id x, I search for the same result in second index (the one with expanded tweets) with tweet id x as a filter and update the final score for that particular document. I weight the scores and give a weightage of 0.7 to the 1st index score and 0.3 to the second index score. After that i get the top N1 results from the second index of expanded tweets and check for the same doc in the first index. If the document is not present i add the new document in the top file, if it is present i move on to read the next result. I kept N1 much less than N because i thought the expanded tweets would not produce as many hits as the actual tweets. This is because when i studied the queries i noticed a lot of queries containing names of people and places which would have been missed out when expanding tweets (Since wordnet does not give any synsets for them). When i expanded the tweets i saw a lot of irrelevant data in the synsets obtained so i gave it lesser weights. Thus if a query has hit on both indexes it would be boosted more. For the filtering task i followed the same procedure as above and in addition for deciding if the tweet has to be shown or not i averaged out the scores per query set to obtain a threshold.

Experimentation and Results :

I tried two analyzers while indexing the tweets alone, the standard analyzer and the english analyzer. The average precision using english analyzer turned out to be better than the average precision for the standard analyzer so i used the english analyzer finally. I also tried putting checks for retweets and language while reading the xml but that resulted on a lot of time spent on indexing so i dropped that approach. Instead i put these check during the searcher phase of the retrieval process. For creating the second index i tried to read each tweet and expand it but that too was very costly. So to save time on that i expanded only queries that were already evaluated in the qrels file. For this reason the second index has only about 73000 tweets. I would have liked to see the map for the expanded whole index but the running time for the program was running upto of 4 to 5 days. Also i studied the effect of parameters when i increased the size of obtaining the result from the second index (size of N1). I observed there was not a significant increase in the number of relevant documents obtained in both AdHoc and Filtering task and thus not significant difference in measurement parameters.  For AdHoc run I tested the results when i obtained first 100 and first 500 results for the second index. For the first 100 results i retrieved i saw the map to be 0.1062. For 500 the MAP is 0.1064 and the number of relevant documents increases by only 5.

Analysis

AdHoc Run:

I have shown below the results of Adhoc run considering only preprocessing done and Adhoc run when i take two indexes. As you can see there is an increase in the number of relevant items retrieved when expanding the tweets. The number of relevant documents fetched increases by 95. However there is only limited numbers that come from the expanded documents index because if we obtain more results from the index there is no significant increase in the number of relevant documents. As you can see in the table below a run of index when we take top 100 from that index gives 1341 results and when we take top 500 gives only 1346 relevant results. An increase in a factor of 5 gives us  an increment of only 5 more relevant documents. Thus i can conclude there is limited scope for applying document expansion using WordNet on tweets. I think the number is very low more so because most of the query tweets had a lot of names of places which had they been in the main tweet would be missed out by the document expansion module. For example one of the queries is “Hugo Chavez”. Now if this occurs in a tweet somewhere the document expansion for “Hugo Chavez” will not expand to give anything because no synsets occur for this word. There are a lot of example for such cases and i think they play a big role in getting not as good performance as expected.

AdHoc Run Results

|  |  |  |  |
| --- | --- | --- | --- |
| parameter | Adhoc Run with Only Preprocessing | Adhoc Run with 2 index approach- Top 100 | AdHoc Run with 2 index approach - Top 500 |
| num\_ret | 52729 | 52816 | 52899 |
| num\_rel | 2572 | 2572 | 2572 |
| num\_rel\_ret | 1246 | 1341 | 1346 |
| map | 0.0974 | 0.1062 | 0.1064 |
| P\_10 | 0.1864 | 0.1915 | 0.1915 |
| P\_20 | 0.1542 | 0.1788 | 0.1788 |
| P\_30 | 0.1435 | 0.1537 | 0.1537 |

Filtering Task:

I tackled the task more or less like i did the AdHoc task and obtained the threshold by averaging out the scores for each document. The experiments obtained very disappointing results for this task. Document expansion did result in getting more relevant results but the number of retrieved documents also increased a lot which resulted in getting lower parameters. On increasing the size of N1 the parameters deteriorated more. I think the deterioration was because of the meddling with scores. Since i weighted the scores and then set a threshold for it i think i cannot discount the possibility that the threshold might have become higher because of new incoming docs from second index. Thus it is possible that the number of documents that cross the threshold would reduce in number.

Filtering run results:

|  |  |  |  |
| --- | --- | --- | --- |
| parameter | Filtering Run with Only Preprocessing | Filtering  Run with 2 index approach- Top 100 - (N1 = 100) | Filtering Run with 2 index approach - Top 500 - (N1 = 500) |
| num\_ret | 23544 | 23800 | 26642 |
| num\_rel | 2316 | 2316 | 2316 |
| num\_rel\_ret | 1239 | 1243 | 1273 |
| set\_prec | 0.1383 | 0.1343 | 0.1131 |
| set\_recl | 0.4866 | 0.4885 | 0.4988 |
| F\_0.5 | 0.1485 | 0.1448 | 0.1244 |
| t11su | 0.0862 | 0.0834 | 0.0685 |

The performance of both the runs were good when compared to other TREC participants. For both the Filtering and AdHoc task the numbers i got would put me somewhere in the middle of the pack in the participant list. I have just explored one operation and its effects. I would like to adopt more methods to see how they will affect the parameters more.