Assignment 1 Report

Sinead Urisohn URSSIN001

Notes and Details of the Implementation

The thesaurus functionality was added as a parameter in *parameters.py* and implemented in *query.py* using the *py_thesaurus* module. To run the code, this module and its dependencies are required. The default parameter configuration of *simple search* is the "control algorithm". *Simple search* extended with the thesaurus functionality, is the "thesaurus algorithm". It was implemented using query-time query expansion. The control and thesaurus algorithms are compared to each other by calculating the precision, recall, MAP and NDCG measurements for each algorithm. The queries for the *emotions_collection* can be run via the *run_testbed_queries.py* file, making sure to specify the use of the thesaurus or not in the *parameters.py* file. The results of these query runs are saved either to "thesaurus_results.txt" or "control_results.txt" in the TREC format. The relevance judgments are also converted to TREC format by the *convert_to_trec.py* file for completeness.

Notes: the code was changed to return the top 30 results or less (if there were less results). The collection was created from Wikipedia documents which conform to the specs in terms of document length and being real documents. The crawled documents can be found in the *emotions_collection* file in the *testbed* folder. The queries, relevance judgments and algorithm run results can also be found in this folder. The *emotions_collection* is saved in a simplified Cranfield Format using only the I, T and W fields. The measurements were calculated using Python code. The code for this can be found in the *calculate_measurements.py* file.

Design of the Implementation

For the thesaurus implementation: at query time the original query is broken up into its constituent terms. Synonyms are then found for each term and the original query is expanded with these synonymous terms. Synonyms that are longer than one word are ignored as the meaning of the original query might not be conveyed with the individual words of a phrase. For the query expansion, only unique synonyms were included, so that there are no repeated terms in the expanded queries. The index is then created for each term and similarities are calculated using accumulators as per the original *simple search*.

The detailed relevance metrics calculations/results.

A graded relevance system of 0-2 was used. A document is considered relevant if the relevance value given is greater or equal to 1 for these calculations. The code for the calculations can be found in the

calculate_measurements.py file. Overall the thesaurus performed better than the control algorithm in all aspects for these particular queries. Recall was increased, this can be seen for query 2, as expected.

Note: For NDCG the ideal was calculate for all 30 documents.

Query 1: love

Query 2: sadness

Query 3: pleasure

The results are summarized below (this is the output of the *calculate_measurements.py* file):

Number Retrieved

Query	Control	Thesaurus
1	30	29
2	6	10
3	30	20

Number Relevant

Query	Number
1	17
2	15
3	9

Number Relevant and Retrieved

Query	Control	Thesaurus
1	17	17
2	6	10
3	9	9

Precision

Query	Control	Thesaurus
1	0.566667	0.586207
2	1.000000	1.000000
3	0.300000	0.450000

Recall

Query	Control	Thesaurus
1	1.000000	1.000000
2	0.400000	0.666667
3	1.000000	1.000000

MAP

Query	Control	Thesaurus
1	0.806112	0.776212
2	1.000000	1.000000
3	0.565027	0.620495

MAP All

Query	Control	Thesaurus
All	0.790379	0.798902

NDCG

Query	Control	Thesaurus
1	0.972109	0.959105
2	0.660905	0.790184
3	0.938305	0.935698

NDCG All

Query	Control	Thesaurus
All	0.857107	0.894996