Cengage Learning at TREC 2010 Session Track

Benjamin King
University of Michigan
2260 Hayward Street
Ann Arbor, MI 48109 USA
benjaminking@umich.edu

Ivan Provalov
Cengage Learning
27500 Drake Road
Farmington Hills MI, 48331 USA
ivan.provalov@cengage.com

ABSTRACT
This paper details Cengage Leaning's TREC 2010 Session track submission and our efforts to improve retrieval performance over a user's session. We use a number of different techniques to achieve this goal including query term weighting, query expansion and re-ranking. In this paper we detail these techniques and the results of our submission. Using our query term weighting technique combined with our corpus term collocation query expansion we were able to achieve 0.2375 for the nsDCG@10.RL13 metric.

1. INTRODUCTION
Our goals were to further research relevance improvements and to work on developing session techniques that might be applicable in Cengage Learning's products. We focused our efforts on two aspects of the problem: 1) creating a strong ad-hoc retrieval system to serve as the framework for our session efforts and 2) implementing a number of various techniques for improving retrieval performance over a session of queries.

2. TASK DESCRIPTION
The goals of the TREC 2010 Session Track are "to test whether systems can improve their performance for a given query by using information about a previous query, and … to evaluate system performance over an entire query session instead of a single query." [7]

Because this is the first year of this track, the task involves a simple session of only two queries, an original query and a reformulation of it. The participants are to submit three runs for each user session, two of which are simply retrievals for the individual queries without considering the session (RL1 and RL2), and the third is a retrieval of the reformulation using the combined evidence from both queries (RL3).

The primary evaluation measure for this task is mean nDCG@10 on RL3. This measure rewards systems with a high-baseline retrieval performance that effectively leverages the evidence of both queries in the session. For this reason, we worked to create a strong ad-hoc retrieval system over the ClueWeb09 collection, evaluating its performance on the 2009 TREC Web Track queries. We optimized our techniques for both ad-hoc retrieval and session techniques to maximize nDCG@10.

3. COLLECTION INDEXING
For this task, we indexed the Category B subset of the ClueWeb09 collection using Lucene[1], a freely available open source information retrieval API. In addition the ClueWeb09 collection itself, in creating an index, we also utilized publically available spam [2] and PageRank[2] scores for the collection. The spam scores were used prior to indexing in order to filter documents that were likely to be spam. Documents whose spam score fell below a fixed threshold were excluded from the index altogether. Among documents that were indexed, we used a combination of the spam score and the PageRank score to endow with an inherent retrieval preference documents that had a high PageRank or were unlikely to be spam.

The index contains four fields:

- Document Text (after parsing HTML)
- Document Title (extracted from the `title` tag)
- URL
- Anchor Text [6]

Prior to indexing a document, we used the Jericho HTML parser[3] to extract the plain text from each document. The Jericho parser is a very robust HTML parser that can correctly handle nearly any type of malformed HTML, though it does, in extreme cases, leave behind some markup in the extracted text. We also performed stemming,

[1] http://lucene.apache.org/

[2] http://boston.lti.cs.cmu.edu/clueweb09/wiki/tiki-index.php?page=PageRank

[3] http://jericho.htmlparser.net/docs/index.html

lowercasing, special character removal, and stop word removal on each document field.

4. AD-HOC RETRIEVAL METHODS

4.1 Retrieval Formula

Our retrieval formula on this collection is a fusion of two other formulas, a modified version of the default Lucene retrieval formula, and BM25F. Using the TREC 2009 Web Track queries and judgments, we have tuned both formulas to achieve their best performance on this collection. To combine the two formulas, we use the expCombSUM fusion procedure, which gave the best performance among all the methods attempted. [8] The fusion retrieval method performed better than either of the individual formulas.

4.2 Query Expansion

4.2.1 Proximity Query Expansion

We augment the queries submitted to the Lucene retrieval formula with Lucene phrase queries and span queries. Phrase queries behave exactly like standard quoted phrases. Because each query potentially contains a number of different phrases, we search for all possible two word phrases in the query (queries much longer than three words tend to be rare, so we believe this is sufficient) For example, given the query "gmat prep classes," the query is augmented with the following phrase queries: "gmat prep," "gmat classes," and "prep classes." [10] Span queries attempt to exploit similar proximity properties. A Lucene span query matches only documents in which the specified search terms (not necessarily in order) are separated by no more than *n* tokens, where *n* is user-specified (we use *n=5*). We augment the query with a span query that attempts to find all the query terms separated by no more than five tokens.

4.2.2 Pseudo-Relevance Query Expansion

Unlike the other expansion methods listed in section 6, pseudo-relevance feedback expansion showed no improvements when gathering expansion terms for the two queries separately, so we ran this method only after we had applied all the other methods. The query was expanded using the most common terms from the top five retrieved documents. Although this method produced considerable improvements in MAP, it actually hurt nDCG@10 in most cases we tested. For this reason, we only used this method in one of our runs.

5. EXPERIMENTAL SETUP

In the absence of any data from prior years with which to test our methods for the Session track, we constructed our own test data set based on the TREC-3 corpus, queries, and judgments. Table 1 demonstrates how additional queries were constructed. For each of fifty TREC-3 queries, we constructed three additional queries, such that there were three query pairs, representing each of the Session track reformulation types (generalization, specialization, and drift). The additional queries were themselves designed to imperfectly represent the information need described in the topic's description, but in combination with the topic titles, to more accurately specify the information need than either query alone. By reflecting the topic's same information need, we were able to reuse the topic's existing qrels for evaluation.

Table 1. Example of query pair construction.

TREC-3 Query Title	dog maulings
Generalization Pair	(pitbull attacks in the US, dog maulings)
Specialization Pair	(animal attacks, dog maulings)
Drift Pair	(dog bites, dog maulings)

Unfortunately, there are some significant differences between our modified TREC-3 environment and the 2010 Session track. The most notable among these are the lengths of the queries and the format of the content. The Session track queries have 2.8 terms on average, while the queries in our test-set contain nearly twice as many, with 5.0 terms on average. The TREC-3 corpus is composed entirely of newspaper articles. The ClueWeb09 collection, being comprised entirely of web pages, contains large quantities of irrelevant text, such as navigational links, copyrights, meta-data, and leftover markup. Nevertheless, experimentation on this test set has proven to be useful, despite the differences between the collections.

Table 2 shows how each of the methods performed compared to the baseline RL1 and RL2 runs. Note that all of the MAP measures are computed against the same set of qrels, so the RL1 numbers may be somewhat inaccurate since the RL1 queries do not always represent the information need well (see above).

Table 2. Performance of various methods on TREC-3 test set.

Method	RL1 MAP	RL2 MAP	RL3 MAP
Term Weighting (Section 6.1)	0.115	0.212	0.241
Category Re-ranking (6.2)	0.152	0.229	0.232
Usage Log Query Expansion (6.3.1)	0.127	0.219	0.220
Corpus Collocation Query Expansion (6.3.2)	0.198	0.238	0.249
WordNet Query Expansion (6.3.3)	0.145	0.236	0.235

6. SESSION METHODS

6.1 Query Term Weighting

In our experimentation, we found that a surprisingly effective technique for improving retrieval performance was to simply apply a new weight to a query term depending upon which query it occurs in. We divided query terms into three different categories: 1) terms that appear only in the first query, 2) terms that appear only in the second query, and 3) terms that appear in both queries.

Through experimentation, we found that the best weights for these three groups were dependent upon what the type of reformulation was (generalization, specialization, or drift). Table 3 shows how the optimal weights differ according to reformulation type.

Table 3. Weights for types of query terms.

Reformulation type	1st query only	2nd query only	Both queries
Generalization	0.5	1.0	2.5
Specialization	0.2	1.0	1.5
Drift	0.3	1.0	0.5

These weights also seem to make sense intuitively, considering the user's intent in each type of reformulation. For example, in the case of generalization, it is more important to remember the first query's terms, whether they are repeated or not, since the second query contains less information.

Of course the requirement to change these weights depending on the reformulation type necessitates the ability to automatically categorize query pairs. Section 7 describes these techniques.

6.2 Category Re-ranking

Category-based (or ontology-based) re-ranking has been explored in literature by a number of different researchers, usually in the context of constructing user profiles for a web retrieval service from user actions, such as issuing queries and clicking on documents. [3] [4] [11] In the Session track, the only pieces of information available for use in constructing such a user profile are the original query and its reformulation. This helps to simplify much of the problem of constructing a user profile, as there is no need to determine session boundaries. However the major drawback is that there is very little information from which to construct such a profile.

Like much of the prior work, we chose to use the Open Directory Project[4] (ODP) as our ontology. The ODP is a massive volunteer effort to manually classify web pages in order to present a comprehensive directory of the World Wide Web. It is freely available to download and is comprised of a large number of categories, each containing a number of web pages that were classified under that category. Each categorized web page has a title, a URL, and a short description.

Our approach here differs from previous work in how we chose the best categories for a query or document. Rather than training classifiers or calculating the cosine similarity between term vectors, we built and searched against an index of all the categories in the ODP ontology. Every category in the ODP was indexed against its title and the descriptions of all the pages categorized under it.

To categorize a query or document, the text of that query or document was submitted to the ODP index and a list of search results was returned with a retrieval score for each result. The top ten results were selected as the best category matches for the query and were given a weight proportional to the retrieval score returned by Lucene.

As this was a re-ranking approach, we were concerned only with modifying the order of the set of documents already retrieved by the previous processes. For each document in the result set, we computed a category match score:

$$sim(d,q) = \sum_{c \in C_q} score(d,c) \cdot score(q,c)$$

where

- *sim(q,d)* is the similarity between a document *d* and a query *q*
- C_q is the set of top ten categories for the query *q*
- *score(x, c)* is the score returned for the category *c* when searching the ODP index for the text in *x*.

To compute the score between a document and the categories, we submitted the full text of the document to the search engine and retrieved a similarity score between the document and each of the top ten category matches for the query.

The approach for re-ranking with respect to two queries was quite similar. Rather than using a single set C_q, we create two sets of categories C_{q1} and C_{q2} which are the top ten categories for each of q_1 and q_2 respectively. The similarity score between a document and a pair of queries then became

[4] http://www.dmoz.org

$$sim\left(d,q_1,q_2\right)=\sum_{c\in C_{q1}\cup C_{q2}} score\left(d,c\right)\cdot\left(\alpha\cdot score\left(q_1,c\right)+\beta\cdot score\left(q_2,c\right)\right)$$

where

- q_1 is the original query
- q_2 is the reformulated query
- α is the weight applied to the original query. We used a value of 0.5
- β is the weight applied to the reformulated query. We used a value of 1.0

To actually apply the re-ranking, we first ranked all the documents in the retrieved set according to their category score. Then using the reciprocal rank fusion method [13] with the original retrieval scores, we computed a final aggregate ranking.

6.3 Query Expansion

While query expansion is a popular technique for improving average precision on a single query, we've found that by using query history to select expansion terms, retrieval performance could be improved beyond that of single-query expansion.

We used several different methods for selecting expansion terms: usage logs, corpus-based collocation, and WordNet relation expansion.

6.3.1 Usage Log Query Expansion

In mining the usage logs from Cengage Learning products, we have produced lists of related search terms. Terms t_1 and t_2 are said to co-occur if 1) they were both searched by the same user in the same session or 2) there was some document d such that searches for t_1 and t_2 both resulted in the user choosing d. [9] Each related term was also weighted according to

$$rel\left(t_1,t_2\right)=co\left(t_1,t_2\right)\cdot log_2\left(\frac{|T|}{|T_{t1}|}\right)$$

where

- $rel(t_1, t_2)$ is the weight given to the term t_2 as an expansion candidate for t_1.
- $co(t_1, t_2)$ is the number of times t_1 and t_2 co-occur.
- T is the set of all pairs of related terms.
- T_{t1} is the set of all terms related to t_1.

This formula is similar in concept to the tf-idf weighting scheme, in that it rewards frequently co-occurring terms, but minimizes the impact of the most common search terms.

Expansion terms were sought for all possible sub-queries, with expansion terms for longer phrases receiving a bonus based on that length. For example, given the query "French Lick Resort and Casino," the entire query itself was unlikely to be found in the usage logs. Shorter sub-queries however, such as "French Lick" or "Resort and Casino" may have had associated expansion terms. Expansion terms for longer sub-queries were given exponentially greater weights based on the number of words in the sub-query. Terms that appeared in the expansions for multiple different sub-queries were given greater weights based on the number of times they appear.

In the combined RL3 run, expansion terms for both individual queries were added. This provided a measurable improvement over expansion on individual queries.

6.3.2 Corpus Collocation Query Expansion

Corpus-based collocation expansion was done very similarly to log-based collocation expansion. The major difference is in how the expansion terms are collected. Cengage Learning maintains a collocation database used for our Search Assist tool. This database was compiled against large portions of Cengage Learning's digital material and can return a list of the fifty most common words and phrases that appear near a given term. Although the corpus has major differences in content and style from the ClueWeb09 collection, expansion using this database has nonetheless proven to be very effective.

We use the same techniques as above of breaking a multiword query into shorter queries to search for expansion terms and of weighting terms according to how frequently they appeared in expansions.

6.3.3 WordNet Expansion

WordNet is a database of word forms and definitions developed at Princeton University [5]. It is organized into sets of synonyms called synsets. Each synset can have a number of different types of relationships to other synsets.

Before we could choose any words for expansion, the words in the query needed to be resolved as to which sense of the word they referred by determining which sense of the word is most similar to the other words in the query. This of course makes the assumption that words with similar senses are more likely to occur in the same query. We believe that this is a reasonable assumption to make. To best resolve the sense of each query term, we use a number of different similarity measures:

- Wu and Palmer [12]
- Extended gloss overlaps [1]
- Tag count frequency
- Part-of-speech matching

The first two methods have been well covered in literature, but the second set of methods, while greatly improving the sense disambiguation performance, do not seem to have been treated thoroughly before.

In our experiments with word sense disambiguation, we found that the senses selected by the well-known similarity measures were often obscure usages that were unlikely to occur in everyday speech or writing. For this reason, we created a measure which incorporated WordNet's tag count, a measure of how often each sense was encountered in the tagging of corpora. Our frequency-based similarity measure does not truly measure the similarity between two senses, but rewards senses that appear with a high frequency. This is also very helpful for single word queries, which have no context with which to resolve the sense. In these cases, we found that the sense with the highest frequency often had the best expansion candidates.

We also found in our experiments that when word forms were ambiguous with respect to their parts of speech, the similarity measures would often select a sense with the wrong part of speech. Using the labels from the OpenNLP[5] part-of-speech tagger, we gave a bonus to senses whose part of speech matched the tagged part-of-speech.

After grouping noun phrases and resolving the senses of all the query terms, we added to the expanded query any words with the following relationships to the query words, weighted according to their tag count:

- Synonym – if a word A has a meaning that is identical to word B, then A is a synonym of B and vice-versa. Example: gregarious and friendly are synonyms.
- Hypernym – if word A is a hypernym of word B, then B is a type or instance of A. Example: fruit is a hypernym of apple.
- Hypernym of a hypernym – if word A has this relationship with word C, then there is some word B such that word A is a hypernym of word B and word B is a hypernym of word C.
- Meronym – if word A is a meronym of word B, then B is composed of or contains A. Example: finger is a meronym of hand.

In empirical testing, expanding the query with these word types led to the greatest increase in MAP.

7. OBSERVED DOCUMENT DISCOUNT

In one of the runs we tried to give the documents that appeared in the first query's top results a discount if they appeared in RL3. This may have been the factor of the improvement for the results for this run when duplicate documents discount metric was used.

[5] http://opennlp.sourceforge.net

8. QUERY CATEGORIZATION

Although it is not required by TREC to identify query pairs as to their reformulation type, we find it useful to produce these labels automatically in order to improve the performance of term weighting, which has different optimal weights depending on the type of reformulation.

We utilized a number of different techniques to attempt to categorize the queries according to their reformulation types. A test on the session track queries, manually classified prior to TREC's official release, indicated that the system correctly classified about 72% of the query pairs.

Each of the following techniques contributes some quantity to a categorization score, which ultimately determines which label the system applies. If the final score is positive enough, the query is judged to be a specialization, negative enough and it is judged a generalization, and too close to zero, a drift.

8.1 Query Term Techniques

A simple and very effective technique for categorizing reformulations is based on observing query lengths and which words appear in both queries. We make the observation that queries with more terms tend to be more specific. If the reformulated query had more query terms than the original, then the categorization score was shifted in favor of specialization by a factor proportional to the difference in query length. (The converse is obviously also true.)

In addition, we also observe that when one query contains all the terms in another query, the first query is nearly always the more specific of the two. When such a case occurred, we added to the categorization score a quantity so large that it was unlikely to be changed unless nearly all the other evidence disagreed with the assessment.

8.2 WordNet Techniques

8.2.1 WordNet Relationships

Some WordNet relationships correlate strongly with the concepts of generalization and specification. We used the following types of relationships to aid categorization:

- Hypernym – if word A is a hypernym of word B, then B is a type or instance of A. Example: fruit is a hypernym of apple.
- Holonym – if word A is a holonym of word B, then A is composed of or contains word B. Example: car is a holonym of wheel. (Holonymy is the opposite of meronymy.)
- Topic – if word A is a topic of word B, then B has its specific meaning only in the context of A. Example: baseball is the topic of pitcher (when referring to an athlete).

From these relationships, we constructed the analog of a hypernym tree (a tree with links for all of the above relationships) for each query term. In order to capture the ideas of generalization and specialization, we attempted to determine when one query term is "above" another in the tree. We settled on this definition: term A is above term B if the two have a common ancestor C such that the distance from A to C is less than half of the distance between B and C. This seemed to identify nodes that have ancestor-descendent relationships while allowing for variations and inconsistencies in WordNet.

Scores were computed for every pair of words in the two queries according to the following equation:

$$relscore\left(q_1, q_2\right) = \sum_{\{(u,v):u \in q_1, v \in q_2\}} relscore\left(u, v\right)$$

where *relscore(u,v)* is 1 when *u* is an ancestor of *v*, -1 when *v* is an ancestor of *u*, and 0 otherwise.

8.2.2 WordNet Definitions

Here we leverage the idea that a word is usually defined in terms of other more general words. If a word from the reformulated query was defined in terms of a word in the earlier query (that is to say, a word from the earlier query appears in the definition of that word), then the categorization score was increased on the specification side. Like the previous technique, we compared all pairs of words in the two queries:

$$defscore\left(q_1, q_2\right) = \sum_{\{(u,v):u \in q_1, v \in q_2\}} defscore\left(u, v\right) - defscore\left(v, u\right)$$

where *defscore(u,v)* is defined as the number of times a synonym of *u* occurs in the definition of *v*.

8.3 Result Set Techniques

Using the intuition that general queries return more results than specific queries, we compared the size of the result sets that the two queries fetched. The original query and its reformulation were submitted to both the Bing[6] search engine and the Lucene ClueWeb09 index. Depending on the ratio of the sizes of the result sets for the two queries, the categorization was made more specific, more general, or left unchanged (if the result sets were too similar in size).

8.4 Query Categorization Performance

When we measured the performance of the query categorization performed manually vs. our system against the prepared queries we found the following. Our manual category assignment agreed 85% of a time with the judges, where our system's categorization agreed 70% of a time. The system performed better in the automatic query categorization for the specialization cases (85%), then for generalization (76%), and fairly poorly for the drifting (49%).

9. RESULTS

We have submitted three different runs, each using a different combination of the methods described in this section. Table 4 shows the methods that were used for each run.

Table 4: Methods used in submitted runs.

Run ID	Methods
CengageS10R1	Term weighting, Corpus collocation expansion
CengageS10R2	Term weighting, Usage-log expansion, Corpus collocation expansion, Pseudo-relevance expansion
CengageS10R3	WordNet expansion, Category re-ranking, Observed Document Discount

Table 5 below shows the results of all three runs. CengageS10R1 had the best performance among the three runs according to the nsDCG@10.RL13 metric.

Table 5: nsDCG@10 performance for RL12 and RL13

Run	**nsDCG@10.RL12**	**nsDCG@10.RL13**	**Difference**
R1	0.2354	0.2375	0.89%
R2	0.2328	0.2347	0.82%
R3	0.2289	0.2294	0.22%

Table 6 shows the metrics that discount duplicate documents between RL2 and RL3.

Table 6: nsDCG@10 performance for RL12 and RL13 considering duplicate documents discount

Run	**nsDCG_dupes@10.RL12**	**nsDCG_dupes@10.RL13**	**Difference**
R1	0.2290	0.2225	-2.84%
R2	0.2260	0.2192	-3.01%
R3	0.2232	0.2227	-0.22%

From the Table 7, CengageS10R3 did better than the other two runs. This table uses nDCG@10 metric.

[6] http://www.bing.com

Table 7: nsDCG@10 performance for RL1, RL2 and RL3

Run	nDCG @10.RL1	nDCG @10.RL2	nDCG @10.RL3	Difference (RL2 and RL3)
R1	0.2176	0.2612	0.2602	-0.38%
R2	0.2146	0.2596	0.2572	-0.92%
R3	0.2094	0.2572	0.2579	0.27%

Finally, Table 8 shows the nDCG and nsDCG for each run by reformulation type. In all three runs the specialization reformulation type's DCG metrics are lower than those of the drift or generalization types.

Table 8: Mean nDCG@10 and nsDCG@10 metrics by reformulation type by each run

Reformulation Type	R1	R2	R3
Drift	0.2679	0.2628	0.2693
Generalization	0.2658	0.2592	0.2588
Specialization	0.1756	0.1793	0.1619
Mean nsDCG@10.RL12	0.2354	0.2328	0.2289
Drift	0.2716	0.2661	0.2704
Generalization	0.2606	0.2529	0.2577
Specialization	0.1827	0.1870	0.1631
Mean nsDCG@10.RL13	0.2375	0.2347	0.2294
Drift	0.2531	0.2475	0.2584
Generalization	0.2524	0.2454	0.2418
Specialization	0.1508	0.1542	0.1316
Mean nDCG@10.RL1	0.2176	0.2146	0.2094
Drift	0.2524	0.2445	0.2416
Generalization	0.2936	0.2937	0.2950
Specialization	0.2412	0.2441	0.2392
Mean nDCG@10.RL2	0.2612	0.2596	0.2572
Drift	0.2473	0.2412	0.2446
Generalization	0.2798	0.2748	0.2890
Specialization	0.2557	0.2574	0.2433
Mean nDCG@10.RL3	0.2602	0.2572	0.2579

10. CONCLUSIONS

In summary, we used a number of different techniques to attempt to improve performance over a user session. The most effective of these were a combination of query term weighting and corpus-based collocation expansion.

None of the improvements were statistically significant. However, comparing nsDCG@10.RL12 and .RL13, we saw some improvement in the CengageS10R1 (less than 1%) with respect to the first goal of using the knowledge of the previous query to improve the results for a given query. The main metric for the second track's goal of evaluating the performance over the entire session (nsDCG@10.RL13) was 0.2375.

We believe that several of these methods may be promising directions for further research. One of our most effective techniques was query term weighting. Our approach was rather simple, dividing the query terms into only three categories and applying weights. There are a number of other similar techniques that may also be effective, such as applying weights based on part of speech, inverse document frequency, or word specificity (perhaps using WordNet).

Another technique which performed well was corpus-based term collocation expansion. The term collocation dictionary was built from our digital content and we are actively researching ways to further improve it.

We also believe that category-based re-ranking could be made to be more effective. This method performed well on our TREC-3 testing set, but we were never able to replicate that performance on the ClueWeb09 collection. We hypothesize that this is because the ClueWeb09 collection is much more diverse than the TREC-3 collection, and is therefore much more difficult to correctly categorize.

Both the CengageS10R1 and CengageS10R2 runs had similar performance as we used similar techniques for both. We attribute the poorer performance of CengageS10R2 to the use of pseudo relevance feedback, which can improve retrieval performance over a large number of documents (*e.g.* n = 1000), but is actually detrimental to the quality of the top documents. Since the DCG measures for this task are evaluated at the 10th result, we hypothesize that pseudo-relevance feedback was primarily responsible for the decrease.

Of the three reformulation types, the specialization reformulation type had the lowest performance when evaluated against the nsDCG@10.RL12 and nsDCG@10.RL13 metrics. Upon further analysis, we found that these two metrics are affected by the lower nDCG@10.RL1 scores. This data shift may be explained by the nature of the specialization reformulation type – the first query is much less relevant to the user's information need than the second one.

Finally, upon a comparison of nDCG@10.RL2 and nDCG@10.RL3, CengageS10R3 shows improvement in overall system performance. CengageS10R3 was least affected by the discounting of duplicate documents because documents that appeared in the first query's result list were discounted in RL3.

11. ACKNOWLEDGEMENTS

We thank Michael Corral, Qiaozhu Mei, Drew Koszewnik, Duane May, Eugene Kiel, Paul Tunney, Rohit Laungani, Pete Pfeiffer, Craig Schroeder, Michael Cafarella, John Nader for their ideas and support.

12. REFERENCES

[1] Banerjee, S. and Pedersen, T. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence* (Acapulco, Mexico), pages 805-810.

[2] Cormack, G. V., Smucker, M. D., and Clarke, C. L. A. 2009. *Efficient and effective spam filtering and re-ranking for large web datasets.* Unpublished Manuscript. http://durum0.uwaterloo.ca/clueweb09spam/spamhunt.pdf

[3] Challam, V., Gauch, S., and Chandramouli, A. 2007. Contextual search using ontology-based user profiles. In *Proceedings of RIAO 2007* (Pittsburgh, USA, May 30 – June 1, 2007)

[4] Daoud, M., Tamine-Lechani, L., and Boughanem, M. 2008. Leaning user interests for a session-based personalized search. In *Proceedings of the Second International Symposium on Information Interaction in Context* (London, United Kingdom, October 14-17, 2008). P. Borlund, J. W. Schneider, M. Lalmas, A. Tobros, J. Feather, D. Kelly, and A. P. de Vries, Eds. lliX '08, vol. 348. ACM, New York, NY, 57-64. DOI= http://doi.acm.org/10.1145/1414694.1414708

[5] Fellbaum, C. 1998. *WordNet, an electronic lexical database*, MIT Press.

[6] Hiemstra, D. and Hauff, C. 2010. *MIREX: MapReduce information retrieval experiments*. Technical Report. University of Twente. http://doc.utwente.nl/71078/1/mirex.pdf

[7] Kanoulas, E., Clough, P., Carterette, B., and Sanderson, M. 2010. Session track at TREC 2010. In *Proceedings of the SIGIR 2010 Workshop on the Simulation of Interaction: Automated Evaluation of Interactive IR.* (Geneva, Switzerland, July 23, 2010), pp. 13-14. L. Azzopardi, K. Järvelin., J. Kamps, and M. Smucker Eds. IR Publications, Amsterdam, 2010.

[8] MacDonald, C. and Ounis, I. 2006. Voting for candidates: adapting data fusion techniques for an expert search task. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management* (Arlington, Virginia, USA, November 6-11, 2006), ACM, pp. 387-396.

[9] Mei, Q., Zhou, D., Church, K. 2008. Query suggestion using hitting time. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management* (New York, NY, USA, 2008), ACM, pp. 469-478.

[10] Mishne, D. and de Rijke, M. 2004. Boosting web retrieval through query operations. In *Proceedings of the 27th European Conference on Information Retrieval (ECIR '05)*, pages 502-516, 2004.

[11] Pretschner, A. 1998. *Ontology based personalized search*. Master's Thesis. University of Kansas.

[12] Wu, Z. and Palmer, M. 1994. Verb semantics and lexical selection. In *The 32nd Annual Meeting of the Association for Computational Linguistics* (Las Cruces, New Mexico, USA), ACM, pages 133-138

[13] Zhang, M., Song, R., Lin, C., Ma, S., Jang, Z., Lin, Y., Liu, Y., and Zhao, L. 2002. Expansion-based technologies in finding relevant and new information: THU TREC2002: Novelty Track Experiments. In *Proceedings of TREC 2002* (Gaithersburg, Maryland, USA, 2002)