The University of Iowa at TREC 2011: Microblogs, Medical Records and Crowdsourcing

Sanmitra Bhattacharya[1], Christopher G. Harris[2], Yelena Mejova[1], Chao Yang[1], Padmini Srinivasan[1,2]
[1]Computer Science Department, [2]Informatics Program, University of Iowa, IA

The Text Retrieval and Text Mining group at the University of Iowa participated in three tracks, all new tracks introduced this year: Microblog, Medical Records (Med) and Crowdsourcing. Details of our strategies are provided in this paper. Overall our effort has been fruitful in that we have been able to understand more about the nature of medical records and Twitter messages, and also the merits and challenges of working with games as a framework for gathering relevance judgments.

TREC Microblog Track

This track addresses the problem of retrieval from a collection of short messages from a well-known microblogging service Twitter. The real-time ad-hoc retrieval task involved finding the "most recent but relevant information to the query" with an emphasis on returning "interesting" but "newer" relevant tweets[1]. The queries presented "an information need at a specific point in time" by specifying the title of the query and also the time it is issued.

This paper describes our four run submissions. Our baseline Run 1 was generated by merging results of several retrieval approaches, with the weakest approach improved by applying a heuristic filter. The final merged result was then cleaned using text-based duplicate detection followed by temporal reordering. Run 2 uses a dataset enriched using external resources, as described below. It is also a merge of several retrieval results – those used in the previous run as well as new ones utilizing new information. Run 3 extends Run 2 by adding query expansion. Finally, Run 4 is the same as Run 3, except for the last temporal sorting of the tweets is skipped, in order to see the effect of "newness" on performance.

Below we describe the acquisition and enrichment of the dataset, our various retrieval approaches, precision-driven filter heuristics, and query expansion experiments. We conclude with a discussion of the performance of our submitted runs.

Dataset Acquisition. Unlike many other TREC datasets, the Microblog collection had to be downloaded from the Twitter server using tools that were provided. This proved to be problematic, since the data on Twitter's servers is dynamic, resulting in differences between dataset across research groups. The problem arose because we were to download tweets based on an ID partially identified by a username. Unfortunately, when a user chooses to change their username, the access path to their previous tweet changes, thus not allowing researchers to download it. This resulted in smaller datasets for research groups that downloaded their datasets later in time.

Dataset Preprocessing. The total number of tweets in our corpus is 15,231,082. Firstly because the track evaluates only tweets in English, we identified the English tweets and tweets in other languages using Python NLTK library[2]. This resulted in 5,857,982 English tweets by 2,681,673 users. We manually conducted error analysis by randomly examining 200 tweets. The result of analysis showed 15 non-English tweets (7.5%) detected as English and 4 English tweets (2%) detected as using other languages. The 7.5% error rate of non-English tweets detected as English appears not crucial, because these non-English tweets are usually ranked lower than English tweets during the retrieval time due to the lack of similarity between them and mostly English queries.

[1] https://sites.google.com/site/microblogtrack/2011-guidelines
[2] http://www.nltk.org/

In order to enrich the information about each tweet, we collected some external information on the URLs and hashtags mentioned. This information was used for Runs 2-4, but not for the baseline Run 1.

1. We expanded the short URL into long URL, and got the webpage title, keywords, and description for every URL. This information was attained using the LongURL API[3].
2. We got the description of hashtags, retrieved from tagdef service[4]. For each hashtag it returns a list of descriptions, in case several are available. Of these, we used the most popular description.

Furthermore, we created two more text-based fields: text reduce (TR) which excludes all URLs, hashtags, and mentions in the original tweet; and text reduce with hashtag (TRH), which excludes only URLs and mentions but includes hashtags. In consequent retrieval experiments instead of using the full message text we use TRH. URLs and mentions are removed as non-content parts that would only introduce noise. We keep the hashtags in the field because they are often used as a part of a sentence, e.g. *#Obama is the president of U.S*, the intent generally being to provide information about the tweet topic.

Initial Retrieval. Two versions of the English part of the collection – one without external information used for Run 1 and with external information for Runs 2-4 – were indexed using Indri[5]. Using logical operations we retrieve several sets of results, ranging from most conservative to least shown below. In each run we use one of three logic operators: QUOTE, AND, and OR, and perform the search on either the whole tweet record (anywhere) or on a particular field: TRH (as described earlier), and URL_{title}, URL_{desc}, URL_{keys} (information gathered on the URLs mentioned in the tweet).

Run 1 (no external resources)	Runs 2-4 (with external resources)
1) QUOTE [anywhere]	1) QUOTE [anywhere]
2) AND [TRH]	2) AND [TRH]
3) AND [anywhere]	3) AND [anywhere]
4) OR [TRH]	4) OR [TRH, URL_{title}, URL_{desc}, URL_{keys}]

The most precise results come from QUOTE and AND queries, but often they do not return any results, or return very few. OR queries, on the other hand, often return a good amount, but of lesser quality. We attempt to improve these using a filter heuristic described below.

Before processing these results any further, though, we remove duplicate documents. We do this using TRH field (which excludes mentions and URLs), since sometimes the same content is retweeted, but with a different short URL (which points to the same webpage).

Capitalized Word Result Filters. Intuitively, we postulate that capitalized words in the query are very important. Manual analysis of initial results supported our hypothesis. Hence, we designed three different strategies to filter out the tweets:

Strategy 1: Filter out tweets which do not contain all capitalized words in query. For example, for query *"Saleh Yemen overthrow"*, both *Saley* and *Yemen* must be in the tweet text.

Strategy 2: The same with Strategy 1 but if the number of capitalized word is only one, take out tweets that do not contain the capitalized words and the right-most non-capitalized word in query; if there is no capitalized word, filter tweets using all possible combinations of words in query (this combination must

[3] http://longurl.org/api
[4] http://api.tagdef.com/
[5] http://www.lemurproject.org/indri/

include the last word). For example, for query *"Texas school robot"*, because there is only one capitalized word *Texas*, we filter out tweets that do not contain *Texas* and the right-most non-capitalized word *robot*. For query *"kepler discovers new planets"* we would require tweets to have either of the following: *kepler* AND *planets*, *discovers* AND *planets*, and *new* AND *planets*.

Strategy 3: Because Strategy 2 is restrictive, if after applying it we get no results, we resort to using the more liberal Strategy 1.

Note that this filter has to be applied to an appropriate field. If we're processing an OR query that was run on URL_{title} field, then this is the field to which we will apply this constraint.

To test these strategies we developed a training set using twelve sample queries provided by TREC. For each, we labeled the top 20 results of the OR queries run on TRH, URL_{title}, URL_{desc}, and URL_{keys} fields resulting in 802 qrel entries. Precision @ 20 and the percentage improvement over baseline can be seen in the Table below.

Table 1. Precision @ 20 for Capitalized Word Filters

Field	Baseline	Strategy 1	Strategy 2	Strategy 3
URL_{title}	0.20	0.54 (+170%)	0.50 (+150%)	0.61 (+205%)
URL_{desc}	0.25	0.38 (+58%)	0.44 (+83%)	0.44 (+83%)
URL_{keys}	0.23	0.55 (+139%)	0.50 (+117%)	0.59 (+157%)
TRH	0.42	0.66 (+57%)	0.57 (+36%)	0.70 (+67%)

The improvement each strategy contributes is significant. Among the three strategy 3 appears the best. However, as discussed later, once the results are merged, strategy 1 performs the best. It is used for our submissions.

Result Merging. There are two merging stages that take place in our runs: first we merge the four OR result sets, and then we merge this set with the more precision-driven QUOTE and AND results. Because Run 1 does not have several OR results, the first merge is unnecessary. First we describe the final merge, being the easier one. The final result set is produced by putting the results of the most conservative runs at the top (QUOTE [anywhere], AND [TRH] and AND [anywhere]), and the rest at the bottom (OR [*]). That is, the final set contains results as ordered in **Initial Retrieval** section. Next for merging the OR results for Runs 2-4 we examined five approaches:

All Or Nothing: take all TRH results, then all URL_{title} results, then all URL_{desc}, then all URL_{keys}. Note the order of the results is determined using each run's performance as estimated in previous experiments, taking results of the best performing run first, then second best performing run, etc.

Round Robin: take one top document from TRH, one from URL_{title}, one from URL_{desc}, one from URL_{keys}, then do another round, etc.

Balanced Round Robin: take three documents from TRH, two from URL_{title}, one from URL_{desc}, one from URL_{keys}, etc. The number of documents taken from each result set is proportional to the difference in performance of each run.

Straight Borda Count [1]: each tweet gets a score, which is a sum of its ranks in all returned result sets. The resulting set is ranked using this score.

Weighted Borda Count: the score is now a weighted sum, using weights reflecting the "goodness" of a run. We use Precision scores attained for each run as weights.

The Table below shows performance of these merging strategies added to each of the filter strategies described in **Capitalized Word Result Filter** section. Note that we are now using Precision @ 30 (official track measure) instead of Precision @ 20 as earlier. The Table also shows MAP.

Table 2. Result Merging Strategies Performance

Strategy	Strategy 1 filter		Strategy 2 filter		Strategy 3 filter	
	P @ 30	**MAP**	**P @ 30**	**MAP**	**P @ 30**	**MAP**
AllOrNothin	0.583	0.359	0.552	0.241	0.640	0.312
RoundRobin	0.647	0.433	**0.555**	0.246	0.643	0.319
BalancedRoundRobin	**0.647**	**0.439**	**0.555**	0.253	**0.643**	0.332
StraightBorda	0.616	0.352	0.552	0.246	0.640	0.331
WeightedBorda	0.591	0.349	0.552	**0.257**	0.640	**0.370**

The best performance in each column is bolded. Notice that comparing the three filter strategies shows that the best is Strategy 1 (compared to superior numbers Strategy 3 showed on individual OR runs earlier). Thus we use Strategy 1 with Balanced Round Robin for submitted Runs 2-4.

Query Expansion. Because of the queries are rather short, we introduced a query expansion step in Run 3. As in the result filter described earlier, we decided to focus on capitalized words in retrieved tweets as potential candidates for query expansion. We use results from QUOTE and AND queries as pseudo-relevant documents (as in, assumed to be relevant to our query). We then rank all of the capitalized words in these documents by frequency of their occurrence. Lexically similar words such as *America* and *American* were detected using Edit Distance [2] and the longer alternatives excluded. We then explored four strategies for choosing expansion terms:

1. select top 1 word ranked higher than the capitalized words in the query
2. select top 2 words ranked higher than the capitalized words in the query
3. select top 1 word which is not one of the capitalized words in the query
4. select top 2 words which are not one of the capitalized words in the query

The selected terms are then added to the original query. For example, training query *"natural disasters Australia"* would have the following list of potential expansion terms (with document frequencies in the brackets): *Yasi* [21]; *Cyclone* [14]; *Australia* [13]; *World* [11]. Using strategies 1 and 3, the new query would be *"natural disasters Australia Yasi"*. Using strategies 2 and 4, the new query would be "*natural disasters Australia Yasi Cyclone"*.

These expanded queries are then used to perform an OR search. Capitalized word result filter is then applied to improve these results. Precision and Recall @ 30, as well as MAP scores for these results are shown in Table 3 (with percentage difference in performance from no expansion results in parentheses). Precision numbers improve noticeably for three out of four strategies. Recall and MAP, however, are hurt. This is expected, since by adding keywords to the query (and the consequent filter), we are improving precision but may disregard documents which otherwise may have been considered relevant. We use Strategy 1 for our Run 3 submission.

Table 3. Query Expansion Results

	No expansion	**Expansion strategy 1**	**Expansion strategy 2**	**Expansion strategy 3**	**Expansion strategy 4**
Precision	0.717	**0.917 (+28%)**	0.911 (+27%)	0.917 (+28%)	0.697 (-3%)
Recall	0.443	**0.401 (-9%)**	0.364 (-18%)	0.401 (-10%)	0.168 (-62%)
MAP	0.405	**0.397 (-1%)**	0.353 (-13%)	0.397 (-2%)	0.161 (-60%)

Time Sort. A final step in the creation of all four submitted runs in a sort of the top 30 documents by published time. Because the main measure for this track is Precision @ 30, we drop results beyond the rank of 30. To examine the relationship between relevance and timeliness, we examined Run 1-3 strategies on our training set. We see that precision and recall @ 30 are not significantly affected, whereas MAP is hurt for all three runs by an average of 9.43% (in the interest of space, we do not include the full table of results).

To explore this relationship further, we reorder results using a linear combination of timeliness (as expressed in number of days from query date) and relevance (as a rank in the result set). Interestingly, we did not find a good proportion for this combination. TREC results below show that skipping the reordering step (Run 4) hurts the precision of our run.

TREC Results. In this section we present performance of our runs as evaluated by TREC organizers. Figure 1 shows Precision @ 30 for the four runs using two relevance sets: all relevant documents (allrel) and only highly relevant documents (highrel). Figure 2 shows the performance of one of our best performing runs (Run 3) compared to the average of median scores reported by TREC across all runs submitted. Unfortunately, at the time of the submission we did not know the requirement that all retweets (the tweets have 'RT' in the text or the response codes of the tweets in the HTML crawl are 302) are considered as irrelevant. Therefore, many of tweets in our result are retweets. We tried to remove all the retweets (the tweets have 'RT' in the text) and generated a new result set. The Precision@30 of allrel improves to 0.305 (7.95% improvement). The Precision@30 of highrel improves to 0.090 (8.54% improvement). Besides, there are still 18% (216/1200) tweets which were not judged because they have response code 302 (they are retweets but there is no 'RT' in the text). So if we eliminate all the retweets, our result would be even higher. In rest of paper, we still use the result set we submitted to TREC.

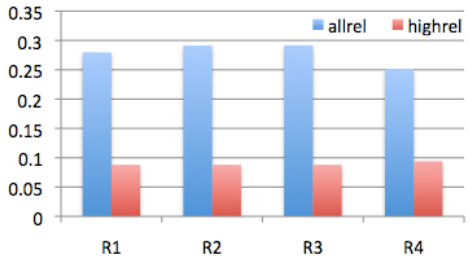

Figure 1. Precision @ 30 for submitted runs

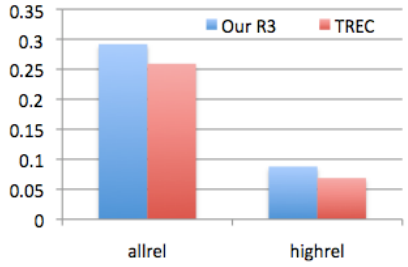

Figure 2. Precision @ 30 for Run 3 and all TREC

Runs 2 and 3 performed equally well, performing slightly better than the average TREC submission. Such a close performance of Runs 2 and 3 suggests that query expansion is not contributing enough benefit. The reason is that the strictness of the expansion strategy allows for only are few queries to be expanded. In the training queries, only 3 queries out of 12 satisfied the query expansion strategy, which requires results of QUERY and AND queries as input (which often return no results). In test queries, only 8 queries out of 50 were expanded. When evaluated using allrel, only 2 queries got better results using query expansion, and the results are the same when using highrel. Hence, we need a strategy that may not achieve the best average precision, but can expand majority of queries with decent precision improvement.

The inclusion of external information (the difference between Runs 1 and 2) does prove to be beneficial. In particular, in allrel, the results of 19 queries have been improved with average improve ratio of 46.4%, but the results of 13 queries have been hurt with the average improve ratio of -3.7%. In highrel, the result of 7 queries has been improved with average improve ratio of 147.3%, but the result of 9 queries has been hurt with the average improve ratio of -58.8% (improve ratio = $(P@30R2 - P@30R1)/(P@30R1 + 0.001) \times 100\%$). Hence, although overall the external information improves precision measures, some queries are negatively affected by it. We leave the study of the contribution of external resources to the retrieval performance for future research.

TREC Medical Records Track (TREC-MED)

The TREC Medical Records Track (TREC-MED) in 2011 addresses the problem of content-based information retrieval from the free-text fields of electronic medical records (EMRs). The corpus for this track consists of de-identified EMRs made available through the University of Pittsburgh BLULab NLP Repository. A typical XML formatted record consists of several tagged elements, namely, *checksum, subtype, type, chief_complaint, admit_diagnosis, discharge_diagnosis, year, download_time, update_time, deid and report_text*. The *checksum* tag consists of a unique identifier for a particular report. The *type* and *subtype* elements contain information about the nature of the diagnosis. The *admit* and *discharge diagnosis* fields contain the International Classification of Diseases, Ninth Revision, Clinical Modification (ICD-9-CM) codes which are used to classify diseases, signs and symptoms and other findings related to health conditions. The *report_text* field comprises of the body of the EMR and was the primary textual element of each report. The other elements of the records are mostly uninformative and irrelevant for the purpose of this task.

One particularly interesting aspect of the task is that multiple reports identified by unique checksum identifiers can be associated with a particular visit identified by a *visit_id*. A mapping table is provided for linking each report to a particular *visit_id*. The *visit_ids* are considered as retrieval units for this task. Similar to traditional TREC tasks, participants are provided with ad hoc query topics specifying a particular disease, sign or symptom and/or diagnosis and treatment. Participants are required to return a ranked list of *visit_ids* that satisfied such queries. Participants are provided with four training queries and corresponding relevance judgments for tuning their system before the release of the test queries. The test topic set consists of 35 queries for which participants had to report a ranked list of *visit_ids*.

The Text Retrieval and Mining group at the University of Iowa participated in the TREC-MED track and submitted four runs (by the original deadline). In the following sections we first describe the general techniques we followed and then provide the more specific information for the particular runs. Finally we present the results for the different runs.

Methods

Dataset. The dataset consisted of 101,711 records identified by unique identifiers (*checksum*). Since the retrieval unit was *visit_id* we mapped the checksum to *visit_ids* using a report-to-visit mapping table. Several reports were not mapped to visits and this gave us a reduced dataset containing 95,702 reports. These reports were mapped to 17,198 unique visits. The number of reports mapped to a particular *visit_id* varied between 418 (*visit_id*: GAgv80e1XdlW) and 1.

Preprocessing. The EMRs distributed for this task contained ICD-9-CM codes in the admit and discharge diagnosis fields which, being numbers, do not convey any information for text-based retrieval. In this step all ICD-9-CM codes of the reports were translated to their literal form using the mappings from the Unified Medical Language System[6] (UMLS) metathesaurus. An index of these translated files was created using the INDRI information retrieval system. As a variant of this, we also merged reports with the same visit id as a single file and indexed those files. Individual fields like admit and discharge diagnosis and report-text were also indexed.

Query Expansion. While training we found that the TREC ad hoc queries contained some words or concepts that do not contribute any useful information for retrieval. Words like patients, treat, take, etc. were present in most of the topics and hence do not convey important and discriminative information. Such words and their variations

[6] https://uts.nlm.nih.gov/home.html

were filtered as stopwords. The training queries were then run through MetaMap [3] to identify important concepts and their identifiers (CUIs). Certain semantic types like "Functional Concept", "Activity", etc. were found to be redundant in the different queries and were removed. After the two-step filtering, only the important concepts of the topics were retained. These were expanded using UMLS following various strategies described in **Submitted Runs** section.

Retrieval Strategies. We explored different query expansion strategies going from most stringent query expansion strategy (using synonyms, narrowly-related terms, etc.) to the least stringent strategy(parent terms, broadly-related terms, etc.). In addition to the high precision Boolean AND queries of synonymous concepts, we also used less stringent INDRI belief operators and window-based matching for improved recall in some queries. Queries were also run on individual fields like *admit_diagnosis, discharge_diagnosis* and *report_text*. These variations in retrieval strategy were run-dependent and are explained in details in the **Submitted Runs** section.

Merging of Results. Since we executed different types of the query expansion and retrieval strategies, for each run we needed to merge the results of several sub-runs. We used the Balanced Round-Robin merging technique as it had given us good performance in other experiments.

Based on the above strategies we submitted four runs which are summarized below.

Submitted Runs:

Run 1: For this run, we execute 3 Indri Boolean AND (*#band*) sub-runs using different UMLS-based query expansion techniques with decreasing stringency in expansion term selection for successive runs. For the first sub-run the filtered concepts were expanded using terms having narrower relationship ("RN"), source asserted synonymy ("SY") and child relationship ("CHD") to the query term identifier (i.e. CUI). For the second sub-run, in addition to the above expansion relationships, we make the range of related concepts broader by incorporating terms having parent relationship ("PAR") and broader relationship ("RB") to the query terms. For the third sub-run we expanded the query terms using terms satisfying all the previous relationship and also similar or "alike" terms ("RL") from UMLS. Other expansion terms having relationship types like "RO" (Other), "RQ" (Possible synonymy) or "SIB" (Sibling) was excluded as they did not perform well on the training topics. The expansion terms are treated as synonyms for the purpose of retrieval. The expansion terms for all the sub-runs were restricted within the same semantic classes as the original query term. Additionally, the source vocabularies were restricted to SNOMED-CT and ICD-9-CM. For the fourth sub-run, which is the least stringent method, we used INDRI belief operator (*#combine*). Unlike the Boolean AND queries which returns only binary values, the *#combine* operator weights each term equally and prioritizes documents containing more query terms. These sub-runs were then merged using the Balanced Round-Robin strategy in a ratio of 3:3:2:2 giving more priority to the two more stringent sub-runs.

Run 2: For this run, we follow the same steps as in Run 1 but here we execute the queries on the index of documents merged on the basis of identical visit ids.

Run 3: For three sub-runs of this run, we follow the same strategies as the previous two runs but here the queries are executed on individual fields like admit diagnosis, discharge diagnosis and report-text only using Indri's *#combine* operator. For the fourth sub-run, the query tries to find all the terms of the query within an unordered window of 4 words. The queries were executed on the same index as in Run 2. Results are finally combined using the Balanced Round-Robin technique in the ratio of 2:3:3:2 giving more priority to topics retrieved in discharge diagnosis and report-text fields compared to topics retrieved in admit diagnosis and windowed matching strategies.

Run 4: This run is similar to Run 1 but results are ranked by first selecting documents satisfying all the terms (or their synonyms) in the query, and then ranking the results according to the #combine of all the query terms (and their synonyms) using INDRI's filter-require operator (*#filreq*).

Results. The results of our four runs in terms of bpref, R-precision (R-prec) and precision at 10 documents retrieved (P@10) are presented in Table 4. Bpref calculates the number of times judged non-relevant documents are returned before relevant documents [4]. R-prec calculates the precision after *R* number of retrieved relevant documents for a query [4]. The evaluations are over 34 topics (topic 130 was dropped).

Table 4: Performance evaluations of submitted runs

Run	bpref	R-prec	P@10
Run 1	0.3619	0.2648	0.3588
Run 2	0.3841	0.2580	**0.4059**
Run 3	**0.4635**	**0.2873**	0.3559
Run 4	0.4131	0.2712	0.3706

From Table 4, we can see that Run 3 performs the best using both bpref and R-prec measures and achieves scores of 0.4635 and 0.2873, respectively. The Run 2 performs the best using the P@10 metric with a score of 0.4059. We are surprised to note that Run 3, where one of the sub-runs had the most basic retrieval operator (unordered window of four words), performed better that other more sophisticated strategies. Also, it is interesting to note that runs executed on the index of files merged using visit ids performed better. Using the same metrices, results for individual topics show that few topics (such as 108 and 109) have performed much better than some other topics (e.g. 110). We are investigating into the reasons for such anomalies. Overall, we seem to have achieved better scores compared to the median scores in terms of the bpref evaluation metric than the R-prec and P@10 measures for the individual topics.

TREC Crowdsourcing Track

Our group participated in the Crowdsourcing track in collaboration with the Delft University of Technology. In this joint participation effort, we employed a term association game called GeAnn to generate relevance judgments in an engaging way that encourages quality submissions from the crowd. A comprehensive explanation of our approach is described in a separate paper [5] written in conjunction with the Delft University of Technology.

Conclusions

In this paper we describe our approach to real-time ad-hoc retrieval in Twitter. Using external resources, we extended the dataset by obtaining more information about URLs and hashtags mentioned in the text. The final results were generated by merging several result sets, coming from approaches ranging from most conservative to more liberal. The less conservative (thus less accurate) approaches were improved using heuristics and query expansion. All of these techniques were tested using a training set of queries with 802 relevance judgments. We show that the capitalized word filter does improve the retrieval precision, but we still have room to achieve better performance. In addition, our query expansion strategy may be too strict to improve performance, calling for a method that can be applied more broadly. Finally, our best submission to TREC outperforms the average precision of all submitted runs, and clearly shows that using external information benefits retrieval.

For the medical records track, we used some basic preprocessing steps to translate the ICD-9-CM codes in the EMRs to textual information. We also explored several query expansion strategies based on the UMLS metathesaurus. While the combination of these strategies gave us some encouraging results, additional work is required to improve the P@10 scores. In future research we would like to find ways to leverage the information content of the report text by using entity recognition methods for identifying standardized biomedical entities.

References

1. Shaw, J. A. and Fox, E. A. Combination of multiple searches. In Harman, D. K., ed. The Second Text REtrieval Conference (TREC-2). (Gaithersburg, MD, August 31-September 2, 1993). The Department of Commerce and the National Institute of Standards and Technology (NIST), 1993, 243-252.

2. Kleinberg and Tardos. Algorithm Design. *Addison Wesley/Pearson*, ISBN: 0321295358

3. Aronson, A. R., & Lang, F.-M. (2010). An overview of MetaMap: historical perspective and recent advances. *Journal of the American Medical Informatics Association*, *17*(3), 229-236. BMJ Group.

4. Buckley, C., & Voorhees, E. M. (2004). Retrieval evaluation with incomplete information. *Proceedings of the 27th annual international conference on Research and development in information retrieval SIGIR 04* (pp. 25-32). ACM Press.

5. Eickhoff, C., Harris, C. G., Srinivasan, P. & de Vries, A. P. (2011). GeAnn at the TREC 2011 Crowdsourcing Track. In *Proceedings of the Twentieth Text REtrieval Conference (TREC-2011).* (Gaithersburg, MD, November 15-18, 2011). The Department of Commerce and the National Institute of Standards and Technology (NIST), 2011.