San Francisco State University at TREC 2014: Clinical Decision Support Track and Microblog Track

Aayush Bhandari, James Klinkhammer, and Anagha Kulkarni
Department of Computer Science, San Francisco State University
San Francisco, California
{aayushb, jamesk , ak}@sfsu.edu

ABSTRACT

The *Clinical Decision Support* Track in TREC 2014 involved identifying biomedical articles that could assist in answering generic clinical questions. This paper discusses the methodologies adopted by the system, Runsystem2, that we built for answering these medical questions. Runsystem2 operates by translating a narrative of a patient's case report into a list of structured medical concepts which are then used to generate the query. The articles retrieved for the query are then re-ranked based on their ability to answer the three types of clinical questions studied in this track: diagnosis, treatment and test. The experimental results demonstrate that the developed system performed close to the median performance on most metrics.

Our approach to the ad-hoc retrieval task focused on on query expansion, language and re-tweet filtering, and URL boosting. Query expansion used pseudo relevance feedback based on frequency of reoccurring terms. The various filters were then applied over the results from the expanded query after which the remaining tweets were re-ranked via URL boosting. The experimental results demonstrate that the best search effectiveness is obtained when all three techniques are employed.

CLINICAL DECISION SUPPORT TRACK

1. INTRODUCTION

The Clinical Decision Support track ran for the first time in TREC 2014. The specific retrieval task defined under this track was that of answering Generic Clinical Questions. The goal is to assist medical professionals by retrieving biomedical articles that answer different types of clinical questions that are related to a particular patient's case report. Three types of clinical questions were included in the 2014 task: diagnosis, treatment or test.

We approached the problem by designing our system Runsystem2 which consisted of three main components: treatment/test score generation, retrieval, and post-processing. Section 2 describes the steps taken to build the index from a provided corpus of biomedical articles using Indri[1]. The assignment of each document in the corpus with a score, based on the relevance of treatments or tests is also illustrated. Section 3 describes the use of MetaMap[2] and MeSH[3] for transforming a patient's narrative into a set of weighted structured queries. Section 4 describes the post-processing step, where documents are re-ranked based on the type of the clinical question/information need. The experimental results are described in section 5. The paper ends with conclusions in section 6.

2. SYSTEM ARCHITECTURE

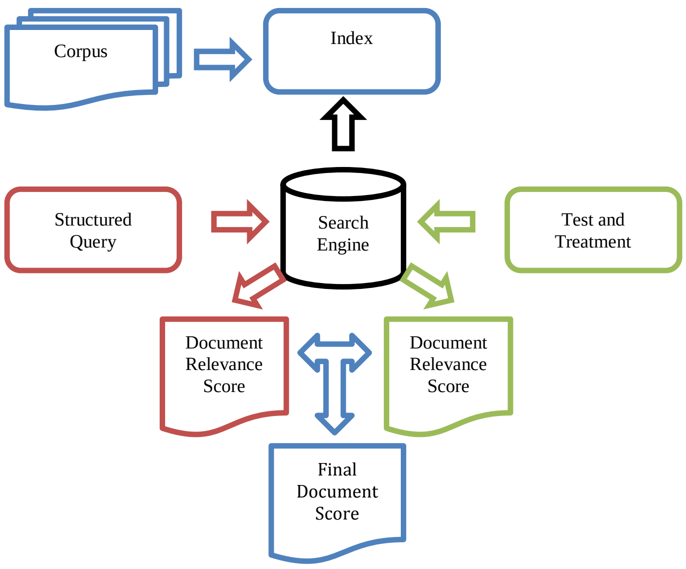


Figure 1: Search System Flow Diagram for Runsystem2

[1] http://www.lemurproject.org/indri/
[2] http://metamap.nlm.nih.gov/
[3] http://www.nlm.nih.gov/mesh/

2.1 Index Building

The 4 PMC-text bundles provided to the participants were transformed into an inverted index for efficient and convenient access using Indri. The krovetz stemmer was employed for morphological normalization. The index creation process was straightforward because Indri supports documents in xml formats. The indexing allowed documents to be identified by their filename, which were the same as the articles PMC Identification. This number also allowed easy access for viewing formatted versions online. In order to perform focused search of certain sections of a document, such as, the abstract, and the article title, which is supported by Indri, some of the fields were included in the indexing process. These fields included abstract, body, article-id, article-title and reference-list.

2.2 Treatment and Test Scores

When studying the document collection one of our observations was that documents that specify one or more treatments for a particular medical condition may not always use the term treatment in the document. We saw similar pattern for medical tests as well. This posed a challenge for the retrieval algorithm when answering treatment and test type questions. The solution we developed for this problem made use of an external resource, specifically a list of 421 treatment names from *rightdiagnosis.com*, and 819 test names from *webmd.com.* Each name from these lists was used as an individual query with the goal of assigning a *treatment score* and a *test score* to each collection document. The fields for searching the treatment and test names was restricted to title, abstract and the body. This field restriction modelled our intuition that a document should be considered relevant to a particular treatment or test only if the key sections (title, abstract, body) mention it, and not if only the references section contains the treatment or test name.

The collection documents were ranked for each treatment and test name individually, using Indri. The relevance score assigned by Indri to each document was then divided by the rank of the document to calculate a weighted score. This score decay operation rapidly dampens the document scores down the list. This was done to obtain a small but robust candidate set of relevant documents for the treatment or test name, which is then used in the final step of post-processing. The weighted scores for a document retrieved by multiple treatment or test queries were summed up to assign a final treatment and test score to such document. Documents that were not retrieved by any of the treatment or test queries were assigned the score of 0 by default.

As a last step the score calculated for treatments and tests were normalized between 0 and 1. Hence, each document in the corpus was assigned with two scores: the treatment score, and the test score.

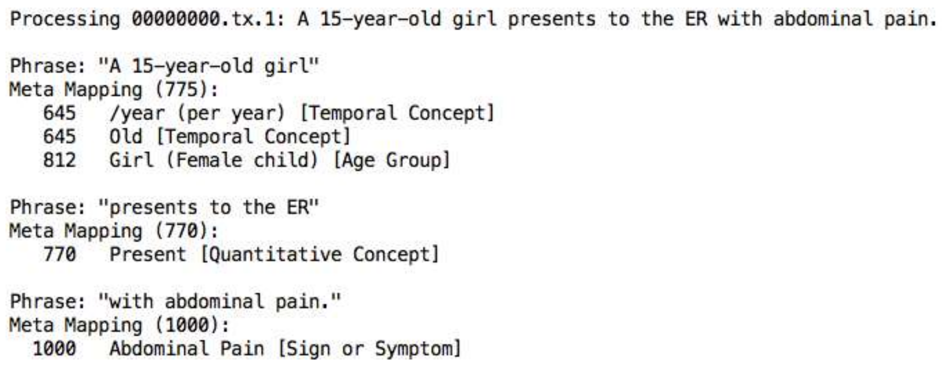
3. QUERY GENERATION and RETRIEVAL

The *Clinical Decision Support Track* provided 30 different evaluation topics. Each topic consists of a summary and a description of the patient's case report. We chose to use the description section of the topic to develop our queries since it contained much more detailed explanation than the summary section.

Upon investigating many biomedical articles we saw numerous medical terms occurring in pairs or triples. Words such as "heart attack" and "myocardial infarction" would make very different sense if they were to appear in a different order, or farther apart. This motivated us to convert the given patient narrative into a set of structured query terms. Upon further inspection we saw cases where instead of being pairs, two or more terms occurred very close to one another. For example the word chest and pain may be portrayed as "chest pain" or "pain around the chest region". To consider these cases such groups of terms each structured term was given term proximity of an ordered or unordered window of length 5. After generating the structured queries each of them were assigned with a weight based on a list of rules described in section 3.2.

3.1 MetaMap Concepts

Recognizing medical concepts in a patient's case report was the most vital step for our system to generate meaningful queries. We decided to use MetaMap, a tool used for mapping biomedical text to the UMLS Meta thesaurus. We employed MetaMap to identify biomedical terms and concepts from the given case report.



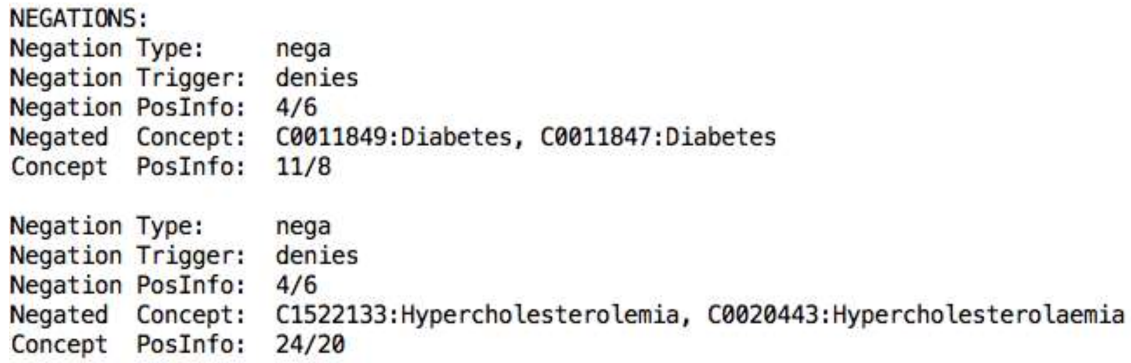
```
Processing 00000000.tx.1: A 15-year-old girl presents to the ER with abdominal pain.

Phrase: "A 15-year-old girl"
Meta Mapping (775):
   645   /year (per year) [Temporal Concept]
   645   Old [Temporal Concept]
   812   Girl (Female child) [Age Group]

Phrase: "presents to the ER"
Meta Mapping (770):
   770   Present [Quantitative Concept]

Phrase: "with abdominal pain."
Meta Mapping (1000):
  1000   Abdominal Pain [Sign or Symptom]
```

Figure 2: Regular output of MetaMap for a phrase

Figure 2 provides an example of of MetaMap's parse and annotations for the phrase "A 15-year-old girl presents to the ER with abdominal pain." MetaMap brokes down the input sentence into phrases, which are then further divided into individual terms or pairs of terms. MetaMap also provides synonyms or hyponym wherever possible and appropriate for the parsed terms. For instance, "female child" is the annotation assigned to the query term "girl" in the above example. The synonyms are specified in round brackets right after the parsed term by MetaMap. One of the most useful pieces of information that MetaMap provides are the semantic type annotations for the parsed terms. These annotations are specified in square brackets. In the above example, "sign or symptoms" is the semantic type assigned to "abdominal pain".

One of the parameters used with MetaMap during this run was -y (word sense disambiguation). The option causes MetaMap to attempt to disambiguate among concepts scoring equally well in matching input text. For example in the phrase "An 8-year-old boy fell from his bike" the word "fell" would be mapped into "fall" and categorized as the season autumn for the Meta candidate with a rank of 1. Using the word sense disambiguation feature allows, "fall" to be categorized to a much more relatable concept of "falling" for the first Meta candidate.

Another MetaMap feature that was utilized was the --negex feature that outputs a list of negated UML concepts occurring in the input and the associated strings that caused the negation.



```
NEGATIONS:
Negation Type:     nega
Negation Trigger:  denies
Negation PosInfo:  4/6
Negated  Concept:  C0011849:Diabetes, C0011847:Diabetes
Concept  PosInfo:  11/8

Negation Type:     nega
Negation Trigger:  denies
Negation PosInfo:  4/6
Negated  Concept:  C1522133:Hypercholesterolemia, C0020443:Hypercholesterolaemia
Concept  PosInfo:  24/20
```

Figure 3: Negex output for MetaMap

In Figure 3 we see the –-negex output of MetaMap for the phrase "She denies diabetes and hypercholesterolemia." Any concepts that were categorized as negations were not included in the query generation process.

3.2 MetaMap Semantic Group Weighting

MetaMap categorizes every identified concept into one of the 133 available semantic types. The semantic types contain concepts such as "clinical drug", "sign or symptom", "population group" etc. The input phrases that are annotated with semantic type that are directly associated with patient's health, such as "sign or symptoms" and "disease and syndrome", were added to the query as phrasal components, that is, the order of the terms and the proximity within terms specified in the input phrase was required in a matching document. Less stringent rules were applied to the input phrases that were annotated with other semantic types. Specifically the order of the terms was not enforced and the terms could be at most 5 terms apart. Input phrases with fewer than 5 terms were added in this manner to the query. The longer phrases were ignored because typically they were not coherent concepts, and also not central to the query topic.

The semantic type was also used to assign weights to the input phrases, based on the following rules. The semantic type of "population group" was given a low weight based on the observation that they are often too generic, and thus would not contribute to answering clinical questions. Similarly "geographic areas" and "language" was also given a weight of zero. Conceptual semantic types such as "temporal concept" and "spatial concept" were give a low weight. For example, the word "right" (spatial concept) in "right arm" would be assigned a very low weight, as the main focus of the concept would be the arm and not which side the arm is in. On the other hand semantic types such as, "disease and syndrome", "sign or symptoms", "body part" were assigned the highest possible weight, as they would be very critical is determining the relevance of a biomedical article. Other semantic types that fell under health, biology and chemistry related topics were given a medium weight. Such topics include, "biological functions", "laboratory or test result" "body system", "neoplastic process" and "mental behavior".

The synonymy and hyponymy annotations assigned to input phrases with highly weighted semantic types were used to expand the input phrase. These expansions were however given one-third weight than of the original concept to avoid query drift.

3.3 MeSH

Query expansion has been shown to be very important in improving retrieval effectiveness in medical systems [6]. MetaMap was able to provide expansion for a few terms but many concepts were still missing expansions. As an alternative method of query expansion, MeSH (National Library of Medicine-Medical Subject Headings) descriptors data's entry terms were used to expand the concepts with the sematic type of "sign or symptoms" and "disease and syndrome". At most 2 top entry terms were chosen for any given concept. For example the concept "syncope" would be expanded with "drop attack" and "fainting". These expansion terms were also structured and assigned with a weight that was one third of the original term to avoid query drift.

3.4 Retrieval

After generating the query based on structure, weight and term proximity, there were two other approaches taken before retrieving the documents. The first step was determining the article field restriction. Abstract and body were chosen as the two fields because they were descriptive and highly likely to cover the article's central topic. Article-Title was not included as a field, since most of the queries contained very generic terms whereas most article title contained very specific medical terms. The reference field was not included because they contained author names and article titles, which were not good representative of the patient's case report.

The second step was using filter requirements, which made it mandatory for a document to contain the prioritized concepts. Most of the highly weighted concepts representing MetaMap semantic group such as "sign or symptom" and "disease and syndrome" were used as the filter terms. Hence, any document that would not contain these terms of the query would just be filtered out.

Using the above described methodology a query was generated for each of the 30 topics. The language modelling and inference network based retrieval algorithm, Indri, was used to run the queries against the collection of biomedical articles, and to obtain a ranked result list for each query.

4. POST-PROCESSING

Upon examining several biomedical articles retrieved for a given patient's case report, we observed that most of the articles provided some discussion about the patient's diagnosis. However, this did not hold true for treatment or test. Hence it was important to filter and re-rank the retrieved documents based on the type of the query topic.

4.1 Final Score Generation

At this point all the documents that were retrieved from the query had three different scores. Indri's relevance score based on the weighted structured query assigned to the document, the normalized treatment score, and the normalized test score computed based on the methodology described in

Section 2.2. For the treatment query type, the relevance score of every retrieved document was boosted by adding the corresponding treatment score of the document. A similar score boost was applied for test query types using the test scores. The retrieved documents were then re-ranked based on the updated scores for both, treatment and test query types. For the diagnosis query types only Indri's relevance scores were used, and thus re-ranking of documents was unnecessary.

	InfAP	infNDCG	R-prec	P@10	Recall
Median	0.0316	0.1514	0.1257	0.2333	
Runsystem2	0.0195	0.1194	0.0926	0.1867	0.2792

Table 1: Runsystem2's search effectiveness, as compared to the median performance.

5. RESULTS

The results are reported in Table 1. For most evaluation metrics the Runsystem2's results are slightly lower than the corresponding median score. Although some queries resulted in a low P@10, the total number of relevant document retrieved was greater, which resulted in a slightly higher average recall.

The query-level analysis for P@10 metric shows that for most queries Runsystem2's performance is close to the median. For some queries, such as, topics 14 and 19, Runsystem2 performed better than the median, but for others, such as, topics 15 and 21, our system did worst. We analyze these queries in further detail in the next section.

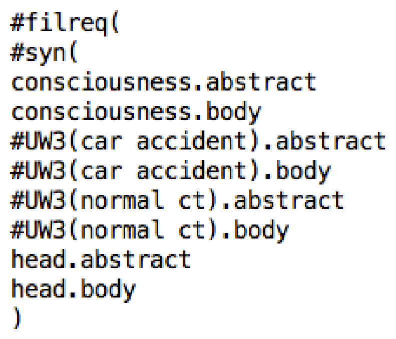
5.1 Query Level Analysis

The queries generated for topic 14 and 19 had very few terms in the filter requirements. The filter requirement for topic 14, *"85-year-old man who was in a car accident 3 weeks ago, now with 3 days of progressively decreasing level of consciousness and impaired ability to perform activities of daily living."* is shown in Figure 4. In contrast, the filter requirements for topic 15 and 21 contained 4 times as many words. The filter requirement for topic 21, *"21-year-old female with progressive arthralgias, fatigue, and butterfly-shaped facial rash. Labs are significant for positive ANA and anti-double-stranded DNA, as well as proteinuria and RBC casts."* is shown in Figure 5. This suggests that too many filter requirements can overly constrain the query and thus degrade its performance. As such, the rules used to add the filter requirements to the query need to be revisited.

Also, the structured query generated by our system for topic 14 and 19 contained few words with high weights, and many words with low weights. The structured query for topic 14 is shown in Figure 6. We believe such a distribution of weights creates a query that is well-focused on a coherent topic (the high weight terms), and also derives support from several related concepts (the low weight terms).

In contrast, the queries generated for topics 15 and 21 consisted of many terms with high weights. The structured query for topic 21 is shown in Figure 7. We believe this type of weight distribution leads to low retrieval performance because the query focus is too diffused and the central topic of the query is not clear. This suggests that it be worth revisiting the term weight assignment rules used by our system, and to allocate high weights to terms more selectively.

Overall, our system is a work in progress, especially so because this is our very first time working with biomedical data and retrieval task. We have identified a couple of promising venues for improvement, and we hope to make further progress once the relevance judgments are made available.

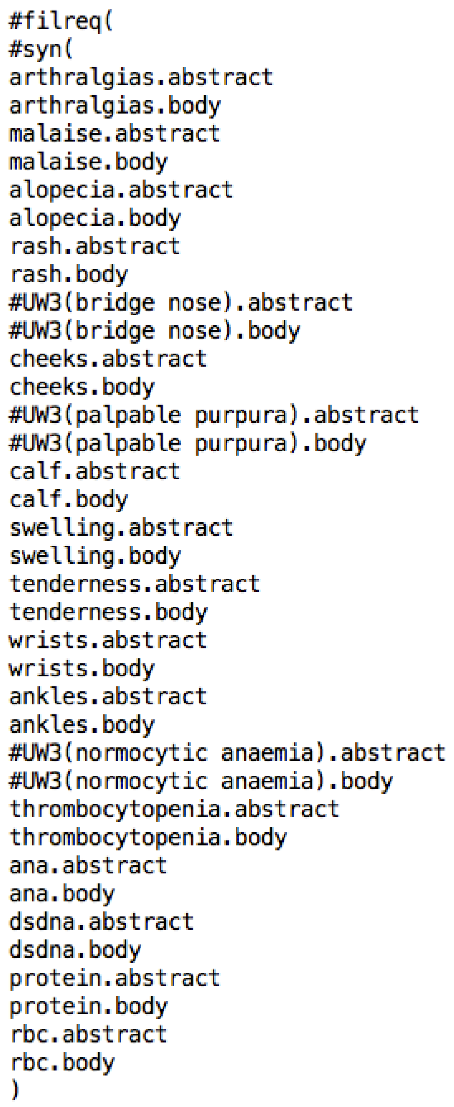


```
#filreq(
#syn(
consciousness.abstract
consciousness.body
#UW3(car accident).abstract
#UW3(car accident).body
#UW3(normal ct).abstract
#UW3(normal ct).body
head.abstract
head.body
)
```

Figure 4: Filter requirements for topic 14

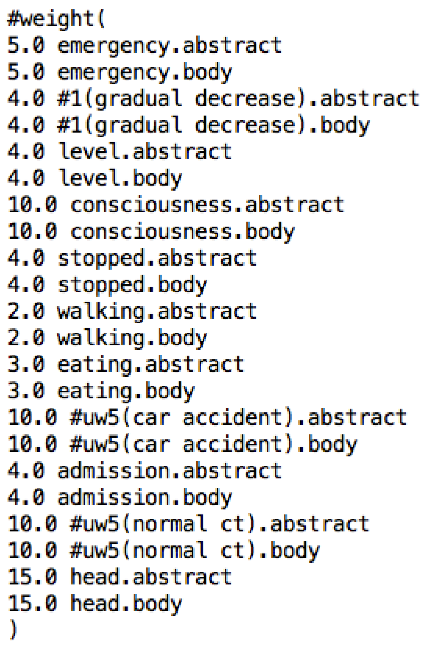
6. CONCLUSION

We presented our system Runsystem2, which uses MetaMap and MeSH to generate a structured query from a case report narrative, and utilizes Indri search engine to find the most relevant biomedical articles for a given patient summary. The documents were further filtered and re-ranked in a post-processing step to adapt the retrieval results to the different types of clinical questions.



```
#filreq(
#syn(
arthralgias.abstract
arthralgias.body
malaise.abstract
malaise.body
alopecia.abstract
alopecia.body
rash.abstract
rash.body
#UW3(bridge nose).abstract
#UW3(bridge nose).body
cheeks.abstract
cheeks.body
#UW3(palpable purpura).abstract
#UW3(palpable purpura).body
calf.abstract
calf.body
swelling.abstract
swelling.body
tenderness.abstract
tenderness.body
wrists.abstract
wrists.body
ankles.abstract
ankles.body
#UW3(normocytic anaemia).abstract
#UW3(normocytic anaemia).body
thrombocytopenia.abstract
thrombocytopenia.body
ana.abstract
ana.body
dsdna.abstract
dsdna.body
protein.abstract
protein.body
rbc.abstract
rbc.body
)
```

Figure 5: Filter requirements for topic 21



```
#weight(
5.0 emergency.abstract
5.0 emergency.body
4.0 #1(gradual decrease).abstract
4.0 #1(gradual decrease).body
4.0 level.abstract
4.0 level.body
10.0 consciousness.abstract
10.0 consciousness.body
4.0 stopped.abstract
4.0 stopped.body
2.0 walking.abstract
2.0 walking.body
3.0 eating.abstract
3.0 eating.body
10.0 #uw5(car accident).abstract
10.0 #uw5(car accident).body
4.0 admission.abstract
4.0 admission.body
10.0 #uw5(normal ct).abstract
10.0 #uw5(normal ct).body
15.0 head.abstract
15.0 head.body
)
```

Figure 6: Structured queries for topic 14

REFERENCES

[1] MeSH Browser. Software. U.S. National Library of Medicine, 1999. Web.

[2] "Medical Tests and Tools A to Z." WebMD. WebMD, 2005-2014. Web. http://www.webmd.com/a-to-z-guides/tests/default.htm

[3] MetaMap - A Tool For Recognizing UMLS Concepts in Text. Software. National Library of Medicine (NLM), 2013.

[4] "TREC Clinical Decision Support Track." TREC, Feb. 2014. Web. http://www.trec-cds.org/

[5] "List of Treatments." RightDiagnosis.com. Health Grades Inc, 17 Jun.2014. Web.http://www.rightdiagnosis.com/lists/treats.htm

[6] Aronson, Alan R. Aronson R.. and Thomas C.. Rindflesch. "Query Expansion Using the UMLS® Metathesaurus®." Print. http://skr.nlm.nih.gov/papers/references/query_expansion.97.pdf

```
#weight(
6.0 progressive.abstract
6.0 progressive.body
10.0 arthralgias.abstract
10.0 arthralgias.body
10.0 arthralgia.abstract
10.0 arthralgia.body
10.0 malaise.abstract
10.0 malaise.body
10.0 alopecia.abstract
10.0 alopecia.body
7.0 rash.abstract
7.0 rash.body
6.0 distributed.abstract
6.0 distributed.body
15.0 #uw5(bridge of nose).abstract
15.0 #uw5(bridge of nose).body
15.0 cheeks.abstract
15.0 cheeks.body
10.0 #uw5(palpable purpura).abstract
10.0 #uw5(palpable purpura).body
15.0 calf.abstract
15.0 calf.body
10.0 swelling.abstract
10.0 swelling.body
10.0 tenderness.abstract
10.0 tenderness.body
10.0 #uw5(sore to touch).abstract
10.0 #uw5(sore to touch).body
15.0 wrists.abstract
15.0 wrists.body
15.0 ankles.abstract
15.0 ankles.body
5.0 lab.abstract
5.0 lab.body
7.0 #uw5(normocytic anaemia).abstract
7.0 #uw5(normocytic anaemia).body
10.0 thrombocytopenia.abstract
10.0 thrombocytopenia.body
4.0 #uw5(positive ana).abstract
4.0 #uw5(positive ana).body
5.0 dsdna.abstract
5.0 dsdna.body
5.0 urine.abstract
5.0 urine.body
5.0 protein.abstract
5.0 protein.body
5.0 rbc.abstract
5.0 rbc.body
2.5 erythrocytes.abstract
```

Figure 7: Structured queries for topic 21

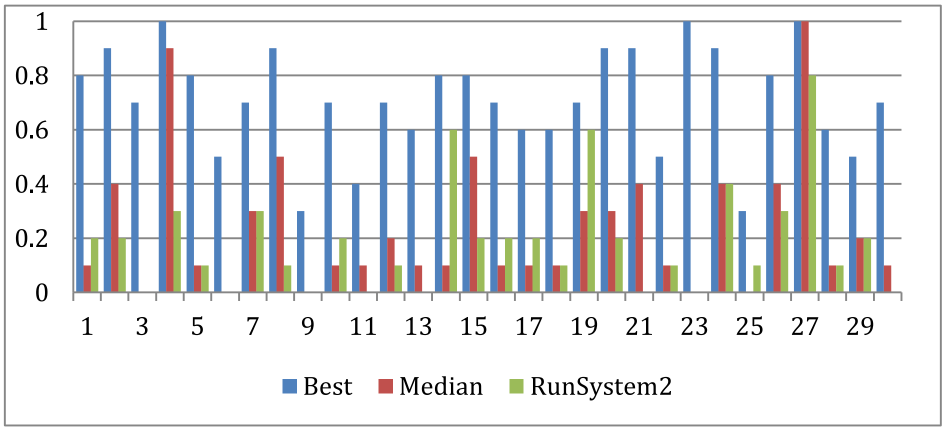


Figure 8: Precision@10 for Runsystem2 and the best & median across all TREC Systems.

MICROBLOG TRACK: Ad-hoc Task

1 Introduction

Microblogs, such as Twitter, are becoming more popular. Their short nature, 140 characters in the case of Twitter, makes searching microblogs unique in comparison to traditional web searches. The 2014 TREC microblog track contains both a temporally-anchored ad hoc search task, and a tweet time line generation task. We chose to participate in only the ad hoc task. The purpose of this task is to retrieve at most the 1000 most relevant tweets to query Q posted before time T. This year's track uses the same corpus as the 2013 microblog track, consisting of 243 Million tweets, accessed through the twitter-tools API [1].

2 Query Expansion

Query expansion involves adding terms in to the original query to improve retrieval effectiveness. Our query expansion method uses a simplified pseudo relevance feedback. First, a baseline search is performed on the corpus using the twitter-tools API and the original queries. The first 20 tweets retrieved for the query are assumed to be relevant. With the help of the Twitter NLP and Part-of-Speech Tagger [2] the proper nouns are extracted from the pseudo-relevant text. All terms that already appear in the query are removed from this set. The three most commonly occurring proper nouns for each query, with a minimum occurrence of 5, are then inserted into the original query. A baseline search with the API is then run using the new expanded queries. We experimented with a range of settings for each of the parameters specified above, such as, the number of top tweets that are considered relevant, and the number of new terms added to the query. The query set and relevance data from the TREC 2011 and 2013 Microblog tracks were used as a tuning set to select the parameter values.

3 Filtering

One of the recurring trends we observed was that the tweets retrieved for the expanded query contained significant fraction of tweets that were of sub-par quality due to one of the following two reasons. We developed simple techniques to identify such tweets, with the goal of removing them from the result list to improve search effectiveness.

3.1 Re-Tweet Filter: Re-tweets are copies of tweets that were originally posted by another user. Upon inspecting the returned tweets we found many re-tweets, beginning with the text "RT@". Because re-tweets are not original content, we presume them to be irrelevant and filter them out by removing tweets beginning with "RT@". For example, a baseline run on the first query (include the query text) in the 2013 query set returns the tweet "RT @anayaseth: whole summer iz left n shortage of water how some1 cn waste water 4 playing holi" as the fourth most relevant tweet.

3.2 Language Filter: While non-English tweets are not necessarily irrelevant, the queries are all English text. While experimenting with the 2013 track's queries, we observed a large number of non-English tweets that were ranked relatively high but deemed irrelevant. Based on this observation, we decided to remove non-English tweets. Using the Language Detection Library for Java [3], developed by Shuyo Nakatani, all non-English tweets were removed from the results. Table 1 shows a sample of the results from the re-tweet filter.

4 URL Boosting

URL boosting refers to increasing the relevancy rank of results if they contain a URL. Tweets containing URLs can contain far more information than just 140 characters and are more likely to provide information relevant to the query than a tweet without a URL. URL boosting has demonstrated consistent improvement of search effectiveness in past microblog tracks [4], [5]. In our experiment, tweets containing a URL had their relevancy rank boosted by a factor of 1.1. We experimented on the 2011 and 2013 queries and data sets with a range of factors between 2.0 and 1.0 and additionally experimented with lowering the rank of tweets not containing URLs. Ultimately A boosting factor of 1.1 was found to provide the best results on the 2011 and 2013 queries. Table 2 illustrates the change URL boosting had on the first query.

id	query	num_rel_ret / num_ret (language filter = off)	num_rel_ret / num_ret (language filter = on)
172	Merging of US Air and American	296/706	296/548
174	Hubble oldest star	13/867	13/734
175	commentary on naming storm Nemo	257/617	252/591
181	Costa Concordia shipwreck	23/733	22/411
183	Evernote hacked	274/808	266/468
185	National Zoo Panda, insemination	22/742	21/562
193	Bulgarian protesters self immolate	90/763	89/667
200	UK passes marriage bill	78/643	78/624
201	Higgs Boson discovery	280/721	277/484
211	Downton Abbey, Lady Mary, beau	16/823	16/782

Table 1. Comparison of the number of relevant results returned and the total results returned with language filter on and off (runs ER and ERL)

Query ID, Top 20 tweets	relevance	Top 20 tweets post URL boost	relevance
299651936842571777	16.79	307496241623883776	16.95
307360182604820481	16.77	302903789118173185	16.94
307462808801513472	16.30	299651936842571777	16.79
307216821277310977	16.22	307360182604820481	16.77
300671001728000000	16.22	307534271374061568	16.44

Table 2. The top 20 returned tweet ids on query MB171 “Ron Weasley birthday” and their relevance rating before and after URL boosting is applied

5 Data

The data set used for the 2014 microblog track is the same set used in the 2013 track. It consists of 243 million tweets taken from the public Twitter stream between February and March of 2013 [6]. The data set cannot be downloaded and is only accessible by running the query set through the twitter-tools API. The difference this year is the query set. The query set consists of 55 different queries varying in length and content. Table 3 summarizes the query set.

query length			number of queries with content relating to:		
Minimum	maximum	average	people/objects	locations	events
2	9	3.76	28	13	35

Table 3. Summarization of the query set. Query content is an approximation. Queries may relate to multiple content categories, or none at all

6 Results and Conclusion

In this paper we have described the methods we used in the 2014 TREC microblog track ad-hoc task. We submitted four runs to the ad-hoc task with various features enabled and disabled. Table 4 depicts the features enabled and the results of each run. Query expansion and the re-tweet filter were both applied to all runs. URL boosting and the language filter were then applied in turn, with the final run making use of all methods. The mean average precision and precision at 30 are the official evaluation metrics for this track. We found that the run making use of all methods performed the best, with the highest MAP and p@30. Using the ER (expansion re-tweet) run as a baseline for ERL and ERU, the language filter had a more significant impact on the p@30 while URL boosting more greatly affected the MAP.

Runtag	Query Expansion	Re-Tweet Filter	Language Filter	URL Boosting	MAP	p@30
ER	Yes	Yes	No	No	0.4013	0.6024
ERL	Yes	Yes	Yes	No	0.4074	0.6170
ERU	Yes	Yes	No	Yes	0.4141	0.6073
ERLU	Yes	Yes	Yes	Yes	0.4200	0.6291

Table 4. Run configurations and comparison

References

[1] J. Lin. Twitter Tools. https://github.com/lintool/twitter-tools

[2] CMU ARK Lab. Twitter NLP and Part-of-Speech Tagging. http://www.ark.cs.cmu.edu/TweetNLP/

[3] S. Nakatani. Language-Detection. https://code.google.com/p/language-detection/

[4] R. McCreadie, and C. Macdonald. Relevance in Microblogs: Enhancing Tweet Retrieval using Hyperlinked Documents In *Proc. of TREC 2013.*

[5] R. Berendsen E. Meij D. Odijk M. Rijke and W. Weerkamp. The University of Amsterdam at TREC 2012 In *Proc. of TREC 2012.*

[6] J. Lin. TREC 2014 Track Guidelines.