Ranking Biomedical Passages for Relevance and Diversity: University of Wisconsin, Madison at TREC Genomics 2006

Andrew B. Goldberg GOLDBERG@CS.WISC.EDU
David Andrzejewski DMANDRZEJEWS@WISC.EDU
Jurgen Van Gael JVANGAEL@CS.WISC.EDU
Burr Settles BSETTLES@CS.WISC.EDU
Xiaojin Zhu JERRYZHU@CS.WISC.EDU
Department of Computer Sciences, University of Wisconsin, Madison, WI 53705

Mark Craven CRAVEN@BIOSTAT.WISC.EDU
Department of Biostatistics & Medical Informatics, University of Wisconsin, Madison, WI 53705

Abstract

We report on the University of Wisconsin, Madison's experience in the TREC Genomics 2006 track, which asks participants to retrieve passages from scientific articles that satisfy biologists' information needs. An emphasis is placed on returning relevant passages that discuss different aspects of the topic. Using an off-the-shelf information retrieval (IR) engine, we focused on query generation and reranking query results to encourage relevance and diversity. For query generation, we automatically identify noun phrases from the topic descriptions, and use online resources to gather synonyms as expansion terms. Our first submission uses the baseline IR engine results. We rerank the passages using a naïve clustering-based approach in our second run, and we test GRASSHOPPER, a novel graph-theoretic algorithm based on absorbing random walks, in our third run. While our aspect-level results appear to compare favorably with other participants' on average, our query generation techniques failed to produce adequate query results for several topics, causing our passage and document-level evaluation scores to suffer. Furthermore, we surprisingly achieved higher aspect-level scores using the initial ranking than our methods aimed specifically at promoting diversity. While this sounds discouraging, we have several ideas as to why this happened and hope to produce new methods that correct these shortcomings.

1. Introduction

The University of Wisconsin, Madison participated in the 2006 TREC Genomics track. The Genomics track investigates how we can design information retrieval (IR) systems that return a diverse set of results based on a user's information need. The participants are given a number of questions such as *"What is the role of PrnP in mad cow disease?"* and are asked to retrieve passages that highlight as many specific aspects of the question as possible, e.g., the psychological impact of PrnP, the neurological impact of PrnP, etc. The participants' submissions are scored in three different ways. First, the passage-level retrieval performance is found: this is measured by the amount of overlap between returned passages and passages the judges deem relevant. Next, the aspect-level retrieval performance is scored by computing how diverse the set of passages returned is. Finally, document-level retrieval performance is computed by essentially counting the number of relevant documents for which a passage was returned.

Our team decided to start out with off-the-shelf components such as the Lemur Toolkit (Ogilvie & Callan, 2001) for our information retrieval needs and focus our efforts on two other aspects: query generation and reranking of query results. The query generation method we implemented uses an in-domain syntactic parser to automatically identify noun phrases in the topic descriptions. Since it is not uncommon in a biomedical setting to have many entity phrases that refer to the same concept, we use online resources to expand our queries with synonyms. Since the goal was to cover as many different aspects of the query topic, our three submissions differed in how we rerank the

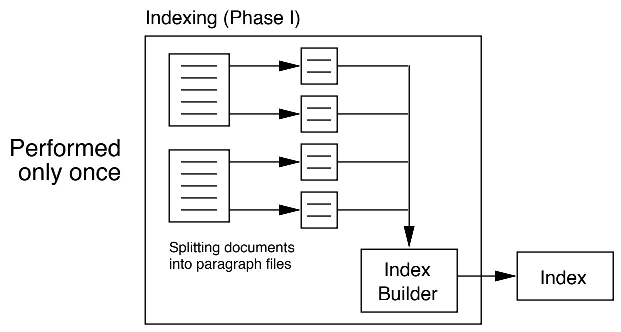

Figure 1. The system's indexing component.

information retrieval results to maximize the diversity of aspects. Our first baseline just uses the order in which Lemur returns the passages. The second baseline naïvely clusters the returned passages and reranks the results by picking out one result from each cluster in turn. Our final experiment uses GRASSHOPPER (Zhu et al., 2007), a novel graph-theoretic approach to reranking information retrieval results. This algorithm uses an absorbing random walk to rerank any set of items to maximize both diversity *and* relevance in a principled way.

The TREC Genomics 2006 submissions are categorized as being generated by automatic, interactive, or manual systems. Groups are responsible for assigning their runs to one of these categories based on the amount of human intervention involved in producing the results. Our three runs fall into the automatic group, as we do not provide feedback or fine-tune any part of the system in response to the quality of the results obtained.

Our system for retrieving biomedical passages from a corpus of documents consists of four primary phases (Table 1). Phase I, depicted graphically in Figure 1, occurs one time only (when the corpus is first obtained), whereas Phases II–IV, shown in Figure 2, proceed automatically for each topic describing a user's information need. Sections 2–5 explore these phases in depth. Section 6 presents the official results of our three runs. Finally, in Section 7, we discuss the strengths and weaknesses of the current system, and describe areas for future work.

Table 1. Four phases of our system.

I Indexing Phase

- Split all documents into paragraphs
- Index paragraphs using IR engine

II Query Generation Phase

- Obtain a topic description
- Identify noun phrases (NPs)
- Find synonyms using online resources
- Build structured query

III Retrieval Phase

- Execute query using IR engine
- Retrieve ranked paragraphs
- Narrow paragraphs into passages

IV Reranking Phrase

- Rerank passages for relevance/diversity

2. Indexing (Phase I)

We decided to use an existing IR toolkit to handle our indexing and query execution needs. Specifically, we used an Indri index built using the Lemur Toolkit (Metzler et al., 2004; Ogilvie & Callan, 2001). Indri combines language modeling and inference networks approaches to information retrieval and provides a powerful structured query language.[1] Lemur provides a framework in which to build an index and use the Indri search engine.

Before building the index, the entire corpus of roughly 160,000 full-text articles from 59 journals was broken up into separate paragraph files using the maximum legal boundaries defined by the TREC-provided "legalspans" file. That is, each individual file corresponds to exactly one maximum legal passage. These separate paragraph files were then indexed by Lemur to form an Indri repository. Note that we did not perform stemming or stopping during indexing.

The pre-processing step of separating paragraphs into separate files has some noteworthy consequences. First, we ignore any document-level information. Separate paragraph files from the same document are handled completely independently. Second, the collection of separate paragraph files contains many files which correspond to non-passage sections of the article, such as references, keywords, and acknowledgments. Empty or otherwise spurious passages will be ignored by the information retrieval system, but some non-passage files may be ranked highly by our information retrieval system. In particular, files corresponding to the keywords section of an article can be ranked very highly due to their high density of relevant keywords, but

[1]A detailed description of the Indri retrieval model can be found at http://ciir.cs.umass.edu/~metzler/indriretmodel.html.

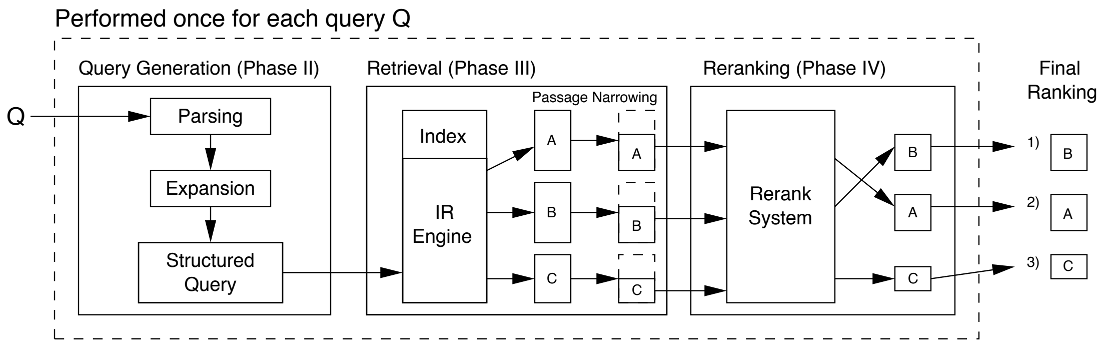

Figure 2. The system's querying components.

these passages would probably not be judged as relevant.

3. Query Generation (Phase II)

3.1. Topic Parsing

One of the goals in our system design is to be able to take topic sentences as input and automatically generate structured IR queries from English natural language text. To do this, we employ an in-domain syntactic parser to identify noun phrases (NPs), and use these phrases as terms in the query. Consider as an example topic 160:

What is the role of PrnP in mad cow disease?

The highlighted words are parsed as noun phrases.

First, topic sentences are tokenized and tagged for part-of-speech (POS) using a modified Brill Tagger (Brill, 1995) trained on the GENIA corpus (Kim et al., 2003). Second, POS output is fed through a shallow phrase chunker implemented with a conditional random field (Lafferty et al., 2001) using the MALLET toolkit[2] trained on the CoNLL-2000 corpus (Sang & Buchholz, 2000) using words, POS, and some orthographic properties such as capitalization as features. We qualitatively compared the results of this simple two-phase chunker on the 28 query topics to the results of a re-trained Charniak Parser (Charniak, 1997) provided by Matt Lease at Brown University for use in this year's TREC task, as well as the Stanford Parser (Klein & Manning, 2003). Our simple chunker appears to produce more sound NPs and runs much faster as well.

[2]http://mallet.cs.umass.edu

3.2. Query Expansion

After obtaining a list of noun phrases in a topic description, the next step in our system is to expand the phrases into lists of synonyms and related terms. Before doing so, we apply a small set of automated heuristics in an attempt to correct any parsing errors and filter out extraneous phrases and stop words. We use the stop list from the Cornell SMART project,[3] but do not filter out single letter stop words, as these may have biological significance. We also include as stop words a small number of common biomedical terms that appeared in past years' topic descriptions (e.g., role, method, gene, etc). Note that if a stop word is detected in the middle of an NP chunk, we remove the word and form two NPs from the remaining words (e.g., a conjunctive NP like "HNF4 and COUP-TF1" is split into "HNF4" and "COUP-TF1"). Returning to the example of topic 160, the first two NPs "What" and "the role" are ignored because they contain common words likely to appear in any scientific query.

Now that we have a set of presumably significant noun phrases ("PrnP" and "mad cow disease"), we expand them into synonym lists by searching the MeSH (Medical Subject Heading) database.[4] We issue each NP as a query to the MeSH Web service and gather the terms associated with the top two MeSH headings returned. We combine these terms with the original NP to form a preliminary synonym list. For each item in this list, we then apply additional lexicographic heuristics to transform the terms into phrases which are more likely to appear as exact phrase matches in a document. Specifically, we remove anything after the first comma (since this is usually some modifier which would not appear in this manner in an actual article). For example, one of the expansion terms for "PrnP" is "prion protein,

[3]ftp://ftp.cs.cornell.edu/pub/smart/english.stop

[4]http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=mesh

human," which we shorten to "prion protein." We also remove parenthetical strings, since these are typically other terms also returned from the MeSH search and will appear in our list separately. Finally, we remove all punctuation, since Indri/Lemur ignores punctuation during indexing.

Based on a technique used in Metzler et al. (2004), we also include in our synonym lists all rare unigram and bigrams within the original NP. We define rare unigrams as those not appearing in a list of the top 2000 most frequent words in the Brown corpus. In the future, we might consider using a more biologically-relevant corpus for such statistics. Applying this expansion technique to "mad cow disease" adds the bigrams "mad cow" and "cow disease," but not the common unigrams "mad," "cow," or "disease." However, for a specialized phrase like "hypocretin receptor 2," we obtain "hypocretin," "hypocretin receptor," and "receptor 2." As a final expansion, we also add copies of words with any trailing 's' removed, in an attempt to convert plurals to singulars. This is a crude heuristic, but it cannot hurt—having an extra synonym which is never found in the corpus will not affect our retrieval results.

For topic 160, the aforementioned expansion techniques produce the following synonym lists:

- **PrnP**: infectious amyloid precursor protein, prnp protein, chromosome 20 amyloid precursor protein, prion protein p27 30, gss protein, prn p protein, sinc protein
- **mad cow disease**: encephalopathy, bovine spongiform encephalopathy, bse, bses, encephalitis, encephaliti, bovine spongiform encephalitis, mad cow diseases, spongiform encephalopathy, mad cow, cow disease

3.3. Building an Indri Structured Query

We utilize several of the Indri structured query language operators in building queries for Lemur to execute. We refer interested readers to the URL listed earlier for a detailed explanation of all the operators and how they are evaluated to compute query likelihood scores.

We describe our query construction through a running example using topic 160. We begin at the level of forming a query term based on a single synonym list. Specifically, we form a #syn term that treats each of the expressions it contains as synonyms. The #syn term contains each item in the synonym list as an exact phrase via the #1 operator. This means we look for documents that contain at least one of the synonyms as an exact match. For example, we represent one of the topic-160 synonym lists as follows:

```
#syn(
    #1(mad cow disease) #1(BSE)
    #1(Bovine Spongiform Encephalopathy)
    #1(Bovine Spongiform Encephalitis)
    ...
)
```

After forming terms corresponding to each synonym list, we combine the synonym lists using the #band operator, which requires all of its operands to be present. For example, we join the topic-160 synonym lists as follows:

```
#band(
     #syn(
          #1(mad cow disease) #1(BSE) ...
     )
     #syn(
          #1(PrnP) #1(prion protein) ...
     )
)
```

So far, our query says that we need to find at least one synonym for each important noun phrase in the topic. The #band requires each #syn to "return true," but this simply means one of the contained phrases must be found.

Finally, we employ Indri's #combine and #filreq operators. Unlike a simple boolean AND, which gives a result of true or false, the #combine operator gives a higher score to results that contain more of its operands. The #filreq operator selects (filters) documents based on one set of criteria (requirements), and then ranks them according to another set of criteria. We assemble these pieces as follows: we use #filreq to first select documents satisfying the #band criteria described above, and then rank the results according to a query term using #combine. The #combine term resembles the #band term, but lacks the #syn operators, thus flattening the synonym lists. We end up with a query of the general form shown in Figure 3.

```
#filreq(
     #band(
          #syn( #1(a) #1(b) )
          #syn( #1(c) #1(d) )
     )
     #combine(
          #1(a) #1(b) #1(c) #1(d)
     )
)
```

Figure 3. General form of the Indri structured queries executed by Lemur to locate relevant paragraphs.

The end result is that Lemur/Indri fetches all the documents meeting the stricter #band criteria, but then ranks them according to how many matching terms are found. If we used only the #band query, Lemur/Indri would essentially rank the documents in increasing length order (due to shorter documents having higher likelihood scores than longer ones).

4. Retrieval (Phase III)

After constructing queries as described above, we execute them against the Indri index built in Phase I. This produces a ranked list of paragraph files satisfying our query, which we map back to byte offsets and lengths within the original documents. We then adjust passage boundaries to include only sentences between the first and last occurrences of key terms from the query. Specifically, we locate the set of consecutive sentences maximally spanning all of the matched query terms. For example, if a paragraph contains sentences A, B, C, D, and E, and sentence B and D contain terms in our query, then we form a passage comprised of sentences B, C, and D.

Consider the concrete example of topic 160. The first result returned by Lemur is the following paragraph, in which we have omitted HTML markup and highlighted the narrowed passage in boldface:

> In December 1984 a UK farmer called a veterinary surgeon to look at a cow that was behaving unusually. Seven weeks later the cow died. Early in 1985 more cows from the same herd developed similar clinical signs. **In November 1986 bovine spongiform encephalitis (BSE) was first identified as a new disease, later reported in the veterinary press as a novel progressive spongiform encephalopathy. Later still the causal agent of BSE was recognized as an abnormal prion protein. Since the outset the story of BSE has been beset by problems.**

The first three sentences lack any exact phrases from our Indri structured query.[5] The next three sentences, however, each contain terms and phrases from our query (e.g., “BSE” and “prion protein”). Thus, we return the boldfaced passage, which is the longest span of complete sentences covering all of the matched terms.

5. Reranking (Phase IV)

Given the narrowed passages obtained in the preceding phase, we next optionally rerank them to promote diversity among the relevant passages and target the aspect-level evaluation metrics.

[5]Our query contained the word “cow,” but only as part of larger phrases.

5.1. Baseline Ranking

Our first submitted run simply lists the narrowed passages in the order in which their containing paragraphs were returned by Lemur.

5.2. Clustering

Our second run naïvely attempts to ensure some amount of aspect diversity through a procedure that begins by performing hierarchical clustering on passage bag-of-words vectors, using a cosine-based distance metric and returning, somewhat arbitrarily, 10 clusters. Under the assumption that clusters group together passages addressing the same topic, we interleave results from each cluster to form the reranked results. We consider the clusters in turn, based on their average initial Lemur ranking. We begin by choosing the cluster whose passages were ranked highest by Lemur. We then remove the highest ranked among them as the first result. Next, we select the second best cluster and remove its highest ranked result. This process repeats until all of the passages are removed from all of the clusters.

The hope is that each cluster represents a distinct aspect, and the interleaving process ensures that a diverse set of aspects is represented high in the ranked list. For example, in topic 160, the cluster-based reranking rearranged the Lemur results to produce the following top five passages (identified by Lemur rank): 1, 9, 27, 3, 2. This means the first result is the same, the second result was ninth according to Lemur, the third result was 27th according to Lemur, etc.

Spot checks after submitting the results reveal that this sometimes produces more diverse highly-ranked results, but often does not. The outcome strongly depends on how reliable the distance metric is, and the quality of the results from Lemur. If some results are irrelevant, they may get ranked highly because they are about a completely different topic than the truly relevant results. This method might have performed better if we could have tuned the number of clusters and selected a distance metric based on training data.

5.3. Ranking for Aspect Diversity

Our third and final run uses the GRASSHOPPER (**G**raph **R**andom-walk with **A**bsorbing **S**tate**S** that **HOP**s among **PE**aks for **R**anking) algorithm to rerank the retrieved passages as to promote diversity. Existing methods to improve diversity in ranking include maximum marginal relevance (MMR) (Carbonell & Goldstein, 1998), cross-sentence informational subsumption (Radev, 2000), mixture models (Zhang et al., 2002), subtopic diversity (Zhai et al., 2003), diversity penalty (Zhang et al., 2005), and others. The basic idea is to penalize redundancy by lowering an item's rank if it is similar to items already ranked. These methods often treat relevance ranking and diversity ranking separately, sometimes with heuristic procedures.

GRASSHOPPER is an alternative to MMR and variants, with a principled mathematical model and strong empirical performance on artificial data. A complete description of the algorithm, and successful results in text summarization and social network analysis, is presented elsewhere (Zhu et al., 2007). For the current task, the algorithm ranks a set of passages such that:

1. A highly ranked passage is representative of a local group in the set, i.e., it is similar to many other items. Ideally, these groups correspond to different aspects.
2. The top ranked passages cover as many distinct groups as possible.
3. The initial ranking from Lemur is incorporated as prior knowledge.

Importantly, the algorithm achieves these in a unified framework of an *absorbing Markov chain random walk*. The key idea is the following: We define a random walk on a graph over the passages. Passages which have been ranked so far become absorbing states. These absorbing states "drag down" the importance of similar unranked states, thus encouraging diversity. The model naturally balances centrality, diversity and the prior. As input to GRASSHOPPER, we use a fully connected graph in which states represent passages. The edge weight between two passage states is based on the cosine similarity of the passages using their bag-of-words representations. Edges between states representing passages with high cosine similarity receive a large weight. After the weight matrix is normalized to form a stochastic matrix, this translates to a high probability that the random walk will move from one passage to another similar passage. If a passage gets ranked and becomes an absorbing state, the similar passages will not be ranked again for several iterations because a walk passing through them will get absorbed.

GRASSHOPPER ends up reordering the topic 160 results considerably, placing the most central passage (i.e., similar to the most other passages) at the top of the list. The top 5 ranked passages are now 141, 16, 11, 15, 35. This means the method placed Lemur's 141st ranked passage as the first passage in the reranked list.

Like the clustering approach, this method is prone to highly ranking irrelevant passages that appear diverse (i.e., not similar to other highly ranked passages). Without training data indicating the aspects associated with example query results, we did not have a good way to evaluate different graph topologies or edge weighting schemes. As a result, it is possible that our graph does not represent the type of similarity relationships (in terms of aspect presence) that we assume exist.

Table 2. Document, passage, and aspect mean average precision scores for the three University of Wisconsin, Madison submissions.

Run	Document	Passage	Aspect
Lemur ranking	0.2368	0.0188	0.1516
Clustering	0.2030	0.0137	0.1319
GRASSHOPPER	0.2208	0.0159	0.1411

6. Results

We now present the results of our runs in terms of the mean average precision (MAP) scores for the document, passage, and aspect levels (Table 2). Mean average precision values are determined by first calculating precision values that represent averages across some unit of text (passage, aspect, or document) for each topic, and then computing the average of these values across topics. While it appears that our document and passage scores are only mediocre, the aspect scores for all three runs appear competitive (compared to the mean of the median scores obtained by all automatic runs). What surprises us most in our results is that the first run (Lemur ranking), which did not do anything specific to promote aspect diversity, actually achieved higher aspect-level scores. We are pleased to see that the more theoretically motivated third approach using GRASSHOPPER did better than the ad hoc clustering-based method.

7. Discussion and Conclusions

We suspect that the poor overall document and passage results are due to inadequate query generation for several topics. In some cases, our topic parsing and expansion techniques failed to produce a set of exact phrases that could realistically be found in journal articles. Consequentially, we obtained few or no results for some topics. One solution would be to relax the exact phrase requirement, using Indri's proximity operators, which would only require the terms to appear within some window. This relaxation could be applied automatically as a fall-back option in cases where the initial query produces fewer than a specified number of results. Of course, the corpus may only contain a handful of relevant passages, in which case we may introduce false positive results.

A better option would be to refine the parsing technique and consult additional resources in search of valid synonyms and related terms likely to co-occur with the terms in the topic description. Some resources we considered using are the Gene Ontology, the Unified Medical Language System (UMLS) Metathesaurus, and the Stanford Biomedical Abbreviation Server.[6] A more traditional approach to query expansion using relevance feedback might also be beneficial. In any case, we could use query term weights to represent our confidence in the various expansion terms depending on their source.

For the topics for which we did obtain numerous results, poor precision scores simply indicates many of the returned passages were deemed irrelevant. In cases where we generated many plausible expansion terms, we often returned keywords or references sections as passages. These are valid spans and loaded with meaningful terms, but it is unlikely that judges would have marked them as relevant.

We are still searching for an explanation as to why our reranking methods actually *hurt* aspect diversity. One possibility is related to the above problems in query generation: we simply did not have a good set of initial passages to rerank. As previously discussed, both the clustering and GRASSHOPPER approaches are prone to placing irrelevant (and thus diverse) passages high in the ranked list. Assuming we did have many relevant passages, the problem with the clustering method lies in a lack of meaningful clusters that group passages by aspect. Of course, the number of clusters is also critical, which probably should depend on the specific set of passages to be reranked. Given relevant passages, GRASSHOPPER strongly depends on a sensible similarity graph that actually captures whether passages share the same aspects. Without aspect-similarity knowledge encoded in our graph, this algorithm will also fail to produce a useful reranking.

To correct these problems, we plan to experiment with alternative passage representations, specifically term frequency–inverse document frequency (TF–IDF) vectors, where the IDF is computed based only on the current set of retrieved passages. We believe this may lead to a cosine similarity measure with greater power in distinguishing passages based on aspects. In addition, we may try other similarity measures, such as the Kullback-Leibler divergence between passage language models (Zhai et al., 2003). We also believe applying a threshold to the similarity measure in order to create a sparser graph may lead to improved results. Finally, we plan to study the behavior of our reranking algorithms when artificially given only truly relevant passages. Separating the reranking phase from the query and retrieval phases will help localize the strengths and weaknesses of the current system.

We should point out that all of the above problems could partly arise from a poor indexing strategy. Indexing complete documents could be more informative than indexing individual paragraphs. While a human judge may be able to determine that a paragraph is relevant without seeing the entire article, this determination may depend on subtle anaphora resolution that the IR engine cannot perform. For example, if a paragraph begins "The disease affects cows' brains by ..." but never explicitly says "mad cow" or one of the phrases in our query, then the paragraph will not be returned as a possible result. Presumably, though, the article included the complete phrase "mad cow disease" or "BSE" in a previous paragraph or the title of the article. Thus, the ability to search at the paragraph level, while making use of document-wide information, is a topic we hope to explore in the future.

We have presented the details of our system and three runs for the TREC Genomics 2006 track. Using an existing IR engine and query language, we concentrated on developing automated query generation techniques, as well as methods for reranking results to boost diversity in the high ranked passages. The methods presented show promise, but still exhibit certain weaknesses that we plan to address in future work.

Acknowledgments

This work was supported in part by a Wisconsin Alumni Research Foundation (WARF) grant. AG was supported by a UW-Madison Graduate School Fellowship. DA was supported by an NLM training grant to the Computation and Informatics in Biology and Medicine Training Program (NLM 5T15LM007359). BS and MC were supported in part by NSF grant IIS-0093016.

[6] http://abbreviation.stanford.edu

References

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, *21*, 543–565.

Carbonell, J., & Goldstein, J. (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. *SIGIR 1998: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. *Proceedings of the 14th National Conference on Artificial Intelligence.* Menlo Park, CA USA: AAAI Press/MIT Press.

Kim, J., Ohta, T., Teteisi, Y., & Tsujii, J. (2003). GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, *19*, i180–i182.

Klein, D., & Manning, C. (2003). Fast exact inference with a factored model for natural language parsing. *Advances in Neural Information Processing Systems (NIPS)*, *15*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 282–289). Morgan Kaufmann.

Metzler, D., Strohman, T., Turtle, H., & Croft, W. (2004). Indri at TREC 2004: Terabyte track. *Proceedings of the Text REtrieval Conference.*

Ogilvie, P., & Callan, J. P. (2001). Experiments using the lemur toolkit. *Proceedings of the Text REtrieval Conference.*

Radev, D. (2000). A common theory of information fusion from multiple text sources, step one: Cross-document structure. *Proceedings of the 1st ACL SIGDIAL Workshop on Discourse and Dialogue.*

Sang, E. F. T. K., & Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking. *Proceedings of the Conference on Natural Language Learning (CoNLL)* (pp. 127–132). Lisbon, Portugal.

Zhai, C., Cohen, W. W., & Lafferty, J. (2003). Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Zhang, B., Li, H., Liu, Y., Ji, L., Xi, W., Fan, W., Chen, Z., & Ma, W.-Y. (2005). Improving web search results using affinity graph. *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Zhang, Y., Callan, J., & Minka, T. (2002). Novelty and redundancy detection in adaptive filtering. *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

Zhu, X., Goldberg, A. B., Van Gael, J., & Andrzejewski, D. (2007). Improving diversity in ranking using absorbing random walks. *Human Language Technologies: Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT).*