Laval University and Lakehead University at TREC Dynamic Domain 2015: Combination of Techniques for Subtopics Coverage

Robin Joganah, Richard Khoury and Luc Lamontagne

Abstract— **This paper describes the joint submission by Laval University and Lakehead University to the TREC 2015 Dynamic Domain track. We submitted four runs for the main track and one run for the judged-only track. In our view, the Dynamic Domain process can be separated into two phases: a traditional static information retrieval phase to generate an initial set of documents, followed by a dynamic information retrieval phase which takes user feedback into account to improve the results. We developed an algorithm that combines keyword search, Latent Dirichlet Allocation, and K-Means clustering to take advantage of each technique to generate the best subset of results for the user. Our systems performed mostly over the median of the group of participants and our system for the "judged-only" task had the best score for a wide range of queries.**

Index Terms—**Dynamic Domain, Information Retrieval, Topic Modelling**

I. Introduction

The Dynamic Domain track challenge (Yang, Frank & Soboroff, 2016) consists in performing information retrieval (IR) in a domain (such as Ebola, Local Politics, and Illicit Goods) using a dynamic process guided by the input of a simulated user. The aim is to return a diversified set of documents that both explores the full range of topics within the subject and researches in greater depth topics the user shows interest in.

We submitted systems for both "main" and "judged-only" tasks. For the first one, we run the system on the whole dataset with some un-judged documents, i.e. documents that can be relevant but we won't have a positive feedback. For the second one, the system only uses judged documents.

In this article, we use the words subtopic as an equivalent to the user's interests in an area. The topic is considered as the initial query. For example the initial query can be "Facebook Accounts" and the subtopics would be "What is the price of a stolen Facebook account" and "Who sells Facebook accounts". Subtopics are hidden in the process and we can only discover them by exploiting the user's feedback.

We investigated multiple approaches in order to retrieve a set of documents that is sufficiently diversified to cover different subtopics. In this paper, we present a method that combines classical information retrieval techniques (using Solr as a search engine framework) with a query reformulation step (in order to construct automatically new queries from the initial search results). We use LDA topic modelling and K-means clustering to select documents and keywords to generate new queries based on the user's interests and original query. By using these unsupervised learning techniques, we aim to create a query that will return both documents in new interesting topics and new documents related to known interesting topics.

In this paper, we will describe how we take user's feedback into account to refine search results, and how we handle the "false start" situation where no relevant documents are returned. We will report on various techniques we used to improve our result set. This includes established algorithms such as Named Entity Recognition (NER) functions and the Rocchio's relevance feedback to extract important keywords. We also report on novel ideas such as alternating between giving positive or negative weight boosts to known keywords in the new queries the system generated. The aim of our system was to allow our algorithm to strike a good balance between exploring the document space of the domain to potentially discover interesting new topics, and exploiting the space of similar documents to potentially discover better documents to represent known topics. Each component of our search algorithm was evaluated, both individually and combined together in a weighted average. Our experiments indicate that a combination of techniques returns the best search results on the training set that was made available for the TREC competition.

II. Related works

Dynamic Domain is a new TREC track, and thus the process is relatively unexplored in the literature. However, the challenge of diversifying IR results taking into account user's feedback has been studied in different forms. One popular technique is to use Rocchio's algorithm (Manning, Raghavan, & Schütze, 2008) which considers the query as a vector with a weight assigned to each word. Then the algorithm updates the weights of the query with words from positives and negatives

documents, words from positives documents having a boost in the vector model, and words from negative documents getting a penalty. Other techniques studied in the literature include clustering (Cohn, Caruana, & McCallum, 2003) and topic modeling (Andrzejewski & Buttler, 2011) to capture diversification in the information retrieval process. Clustering is mostly used to be able to group documents in a fixed number of categories. Topic modelling has been used to find the most probable words for each topic and be able to formulate a new query from these words.

III. Global view of the system

We can separate our system in two phases. In the initial search phase, the system only has the original query and the IR process returns five documents for this query. The second phase is the dynamic refinement phase, during which the user's feedback about the documents is used to reformulate the query, and this new query is used to return more documents. These two phases are illustrated in Figure 1.

The Dynamic Domain challenge can be considered a low-recall task because of the important amount of irrelevant or un-judged documents. In order to get useful feedback from the user, it is important to generate a good set of initial documents in the initial phase.

A. Initial Search Phase & Information Retrieval Process

The first phase consists in an information retrieval process to obtain an initial set of five documents based on the user's original query. This information retrieval process uses the popular Solr search engine to retrieve a set of 20 documents from the dataset. A classic IR system like Solr can retrieve the top documents by keywords, but it doesn't provide the diversification of results needed for the Dynamic Domain challenge; for this, it is necessary to discover topics present within the result set and to return one representative document per topic. We experimented with two algorithms for this purpose: we used a K-means clustering to discover clusters of topics within the search results (Steinbach, Karypis, & Kumar, 2000), and a Latent Dirichlet Allocation (LDA) algorithm (Andrzejewski & Buttler, 2011) for topic modelling. These additional algorithms allow us to re-rank the documents and select the five best results to return to the user.

B. Dynamic Refinement Phase

The dynamic refinement phase starts with a set of five documents provided to the user, either from the initial search phase or from a previous iteration of this refinement phase. The user provides feedback to the system, by indicating which documents and which passages they find interesting. This information, along with the topic's title, allows our system to reformulate the query to find additional relevant documents given the user's interests. We used Rocchio's algorithm to extract keywords for query reformulation, given its demonstrated capability to improve information retrieval results (Salton & Buckley, 1997) and to work on the query-vector model (Manning et al., 2008). To further help the algorithm recognize the most potentially informative keywords, we paired it with a Named Entity Recognition (NER) algorithm.

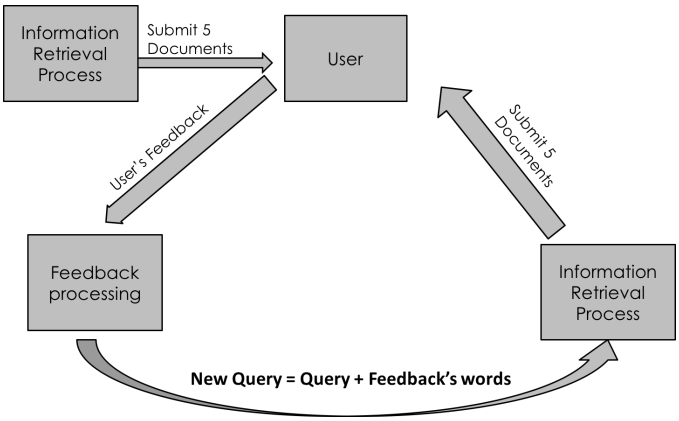

Fig. 1. Global System Overview

C. Halting Condition

As illustrated in Figure 1, phase 2 of our system is an endless loop. The system has to decide when to halt the search and break out of this loop, when an additional iteration will no longer yield useful refinement of the search results. We experimented with three different halting strategies. The first one consists in hard-coding a two-iteration limit, which means that the system would only consider the feedback from the user for one iteration of the loop. The second strategy consists in hard-coding a three-iteration limit. Finally, the last strategy is to use at least 2 and up to 10 iterations, and to have the algorithm stop if it found no relevant documents during the last iteration. We will compare these strategies experimentally in Table 1.

IV. Information Retrieval Process

The information retrieval process occurs at two moments in our system. The first time is during the initial search phase, discussed previously, where it must generate an initial set of five documents based on the user's original query. It then occurs during each iteration of the dynamic refinement phase of the system, to obtain a new set of five documents using the generated query. This IR process has an impact on the performance of our entire system, since the Cube Test (CT) and μ-ERR metrics used to evaluate the TREC track penalize sessions with irrelevant documents as a waste of user's time.

In order to get the best set of five documents for a query, we have to get a balance between topic diversification and specialization (determined by search engine scoring). Diversification is very important for this task because a lot of documents are redundant copies from web pages and news articles; as a result, the five most relevant documents for a query could all contain exactly the same information. We implemented diversification by using a K-means algorithm and an LDA algorithm for topic modeling. Nonetheless, the Solr search engine scoring provides documents that are the most relevant to the initial query and thus to the user's needs. For the final set of results, we thus take a weighted average of these three algorithms' results. The complete IR process is illustrated in Figure 2.

A. Solr

We use the Solr search engine as our IR baseline. We used it to retrieve documents given a query, either the user's original query or the reformulated queries generated by our dynamic refinement stage. The top five documents returned are kept as the Solr recommendation, the five most relevant documents given the query. Meanwhile, the top 20 documents returned by Solr are used as a corpus for the LDA and K-means topic modeling algorithms. These two algorithms implement the complementary diversification of the results.

B. Latent Dirichlet Allocation (LDA)

LDA is an algorithm that takes in a distribution of words and discovers the groupings of topics it implies. In our system, we set LDA to the task of discovering five different topic groups from the top 20 documents returned by Solr from the current query. Next, the system builds five expanded queries, one for each of the five topics groups, by adding the five most probable words in the distribution of each topic group to the current query. It then runs a new Solr search with each of the five new queries, and keeps the top document of each one as the set of five LDA recommendations.

C. K-Means

K-Means is an unsupervised clustering algorithm. In our system, we use it to create five clusters of documents from the corpus of 20 documents returned by Solr, using the documents' word vectors and cosine distance for similarity. The system keeps the document closest to each cluster centroid to create the set of five K-means recommendations.

D. Combination

After these steps, our system has generated three sets of five recommended documents given a query. Moreover, these sets can contain some documents in common. We thus decided to create a weighted combination of these three sets. For the "main" task of the TREC competition we used a repartition of weights of 49%, 25.5%, and 25.5% for Solr, LDA and K-Means results respectively. This gives more importance to a document selected by Solr as relevant to the query over documents selected by either one of the diversification techniques, but more importance to documents selected by both diversification techniques simultaneously as a good representative of an additional topic. For the "judged-only" task of the TREC competition, we found that LDA produced better results than either Solr or K-means individually. Consequently, we changed the weights to 30%, 45%, and 25% for the Solr, LDA and K-means results respectively.

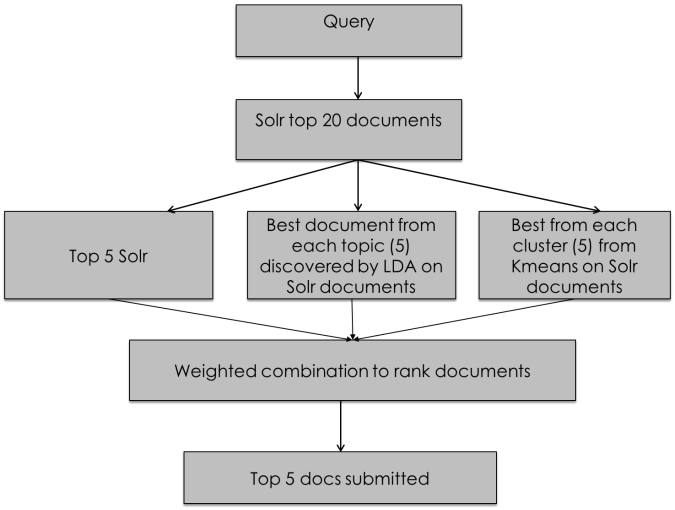

Fig. 2. Information Retrieval Process

V. Dynamic Refinement Process

The dynamic refinement process of our system takes in the user's feedback on the set of five documents retrieved and uses that feedback to improve the results. There are two possible situations the system could be in. The first is if the user marked at least one document as relevant; we can consider that we are in the good area of the domain to search. The second situation is the "false start" situation, when there are zero relevant documents in the set of five results.

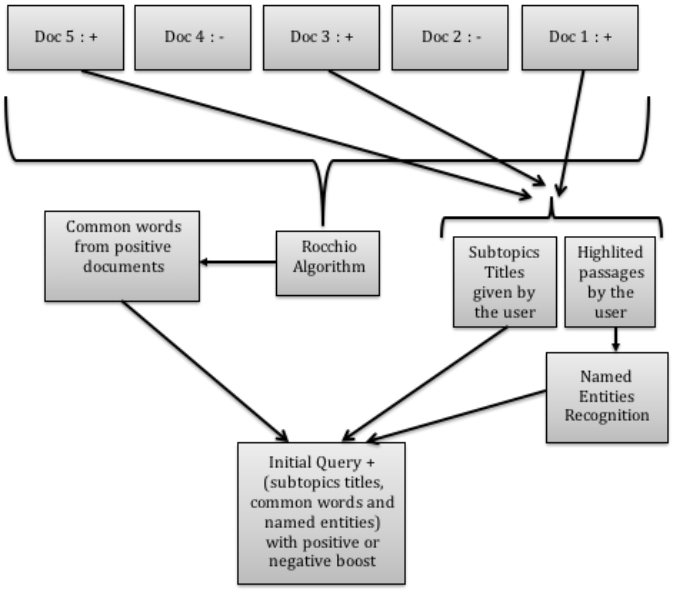

Fig. 3. Feedback processing

In the first situation, our dynamic refinement algorithm will take advantage of the positive feedback information to generate a new query. Our algorithm has access to different sources of information, including the topic title and the passage of the document marked as relevant by the user.

We used the extraction of title's words as we considered as a high informative part of the feedback. Because of the important difference between passages length, taking the whole passage can't be an option in most situations.

In order to focus on the most informative parts from the passage, we decided to use Named Entity Recognition from NLTK to capture named entities on it. We considered that if something is named in the passage, it means that this entity can have a relative importance or meaning for the author for the related subtopic.

We also extracted words with Rocchio's algorithm applied on our five whole documents. This algorithm works by applying a negative weight for terms present in negative documents and boosting words from positive ones. With a TF-IDF vector model we can extract common and informative words from positive documents. With these new words we can compute a new query, which will be used on our static process as the initial query to provide five new documents to the user.

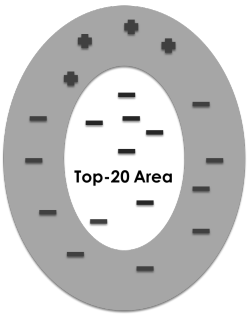

Fig. 4. False-Start

In the second or "false start" situation, the user has marked none of the five documents previously returned as relevant. This situation is difficult for two reasons. The first one is that we are in a bad area, and it might be difficult to find a relevant document. The second one is that we didn't found document and we lost some time in the process of retrieving documents. The CubeTest metric take time into account and, as a consequence, we have to find relevant documents quickly to perform better. The false-start means the algorithm has found itself in a bad area of the domain space, as illustrated in Figure 4, and it is critical to jump to a different area. This is implemented by using Rocchio's relevance feedback algorithm on all five documents to extract their most common keywords. These keywords are given a negative boost in a new Solr query, in order to retrieve very different documents in the next IR process. We found that this approach yields better results than simply taking the lower ranked documents of the previous query.

VI. Experimental Results

A. Data

The dataset proposed for this competition is separated in three domains.

1) Ebola

The Ebola dataset is related to the Ebola outbreak in 2014-2015, it contains 497,362 web pages.

2) Local Politics

The Local Politics dataset is related to related to local politic from Pacific Northwest. It contains 6,831,397 web pages from October 2011 to February 2013.

3) Illicit goods

The last dataset is related to how illicit goods are sold and advertised on the web. It contains around 3 millions post and 526,717 threads from hacking forums.

B. Data Pre-Processing

We pre-processed the corpus by using the porter stemmer algorithm over the whole corpus with the library NLTK and we used the lemmatization from WordNet. We also cleaned the html tags and punctuation with NLTK.

C. Metrics

1) Cube Test (CT)

The Cube Test metric (Luo, Wing, Yang, & Hearst, 2013) uses the analogy of a empty cube to fill with information on each subtopic. This metric has a maximum of 1 as the height of the cube. It is defined as:

$$CT(Q,D) = Gain(Q,D)/Time(D) (1)$$

where Q is the information need and D the list of document. The metrics works as the measure of the changing volume during a period of time. The higher the Cube Test is, the better the system is.

The fact that the Cube Test take into account the period of time indicates that if we take multiple "turns" to find documents, the score will be penalized instead of finding early documents and stopping the search in the early turns.

2) μ-Expected Reciprocal Rank (μ-ERR)

μ-ERR is a variant of Expected Reciprocal Rank (Chapelle, Metlzer, Zhang, & Grinspan, 2009) which calculates the ERR for each topic and averages the scores across subtopics arithmetically.

It is defined as:

$$ERR := \sum_{r=1}^{n} \frac{1}{r} P(user\ stops\ at\ position\ r)$$

where n is the number of documents in the ranking.

D. Training Results

1) Results for Main and Judged-only Tasks

We used the TREC data provided in order to train and fine-tune our system. We compared each IR algorithm alone and in combination to see which one performed better in the main task and in the judged-only task. The results are presented in Figures 5 and 6.

In both cases, it can be seen that LDA performs better than the other algorithms for the CubeTest metric, while Solr performs better for the μ-ERR metric. This illustrates the need for a combination of all three algorithms as we have used, to match up the strengths of each and compensate for their weaknesses. Indeed, our results show that the combination of algorithms yields, in the main task, a performance almost equal to LDA on the Cube Test metric and almost equal to Solr on the μ-ERR metric. In the judged-only task, the combination of algorithms actually yields the best performance on the Cube Test metric, but at the cost of a lower second-best performance on the μ-ERR metric. This improvement in the results comes from the additional

information available to the combination algorithm. A document retrieved by all three algorithms has a better chance of being relevant than a document retrieved by two algorithms, and that one still has a better chance of being relevant than a document retrieved by one algorithm alone.

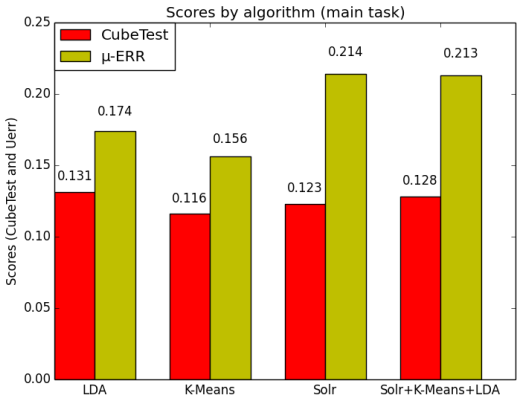

Fig. 5. Results for "main" task

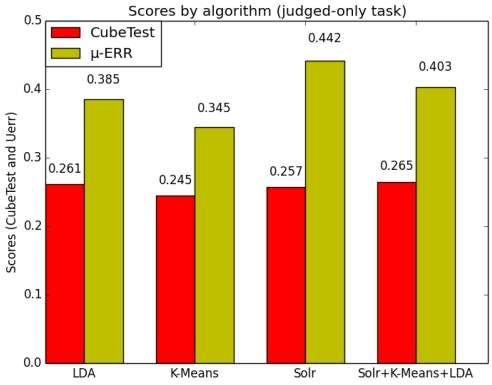

Fig. 6. Results for "Judged-Only" task

2) Feedback Analysis

We have different options when we want to use the feedback, we can use words from relevant documents to exploit our knowledge of discovered subtopics to find other relevant documents for related subtopics. Or we can make the assumption that we want documents that have different words from already discovered subtopics and try to give a negative boost to words from our feedback.

To study how to use the user's feedback in the dynamic refinement phase of our system, we start by separating the feedback into two groups of words: (1) the words extracted from portions of the text (text passage with NER algorithm and topic Title), and (2) the words extracted with Rocchio's algorithm from whole documents. We gave each group alternatively a positive (+) or negative (-) boost in the new Solr query. The goal is to see how these differences affect the results (Figure 7).

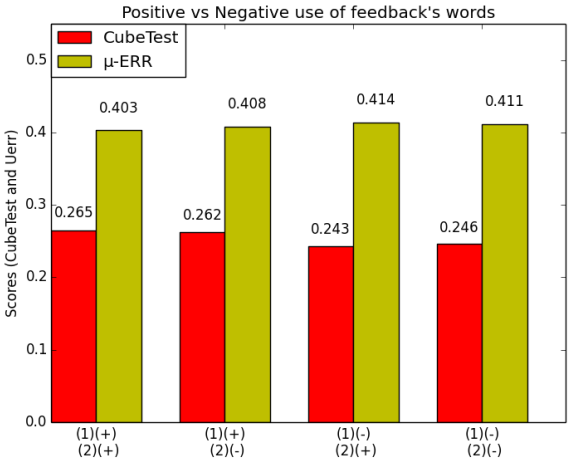

Fig. 7. Results of comparison between positive and negative feedback processing

We can see that giving a positive boost to the text-portion keywords yields better results for the Cube Test metric than using a negative boost. We can explain this result by the fact that words from topic's title and text passage's can be shared between different topics. Therefore, excluding them can lead to a loss of information. However, giving these words a negative boost has a positive impact on μ-ERR metric, which means the ranking of the documents becomes more accurate.

TABLE I
SUBMISSIONS RESULTS

ID Submission	Task	Algorithm	Weight	Best results (ct@10) % (of topics)	Over Median (ct@10) %	Best results (μ-ERR) %	Over Median (μ-ERR) %
ul_lda_roc.2	Main	LDA only	None	2.5	58.5	2.5	20.3
ul_combi_roc.2	Main	LDA + K-means + Solr	0.255 + 0.255 + 0.49	0.8	60.2	4.2	18.6
ul_lda.roc.3	Main	LDA only	None	0	53.4	1.7	22.0
ul_lda.roc.10	Main	LDA only	None	0	6.8	2.5	21.2
ul_combi_roc_judged	Judged-Only	LDA + K-means + Solr	0.45 + 0.25 + 0.30	39.8	98.3	20.3	76.3

VII. TREC Submissions Results Analysis

The "main" task and the "judged-only" task have to be analyzed separately. One part of the competition involved 6 million documents, while the other part was performed only with 25,000 documents. The first corpus includes un-judged documents, which appears to have no feedback but could be relevant to a topic. As a consequence, it adds an important amount of noise for the algorithms. The second corpus only have judged-documents. The information provided by the feedback is more reliable, because we can not have false-negatives documents.

For the "main" task, we evaluated four different system configurations. We submitted three evaluations with it, using the three halting conditions described previously with LDA. However, a system comprising a combination of techniques was able to perform better in some way with more balanced results between CubeTest and µ-ERR metrics. As we can see on the first and second row of Table 1, LDA can provide better results for the Cube Test and lower results over the median than the combination system. On the contrary, the combination of techniques has more results over the median and lower percentage of best results. For the µ-ERR this result is exactly at the opposite.

These results can be explained by the ability of LDA to do topic modelling and to be able to capture more topics. By doing a good diversification, the topic modeling process is also able to avoid more false starts. However, the diversification of documents retrieved by Solr only is not enough. As we saw previously, K-means alone performed poorly. Instead of only diversifying documents retrieved by Solr, the topic modelling allows to create a new query with extra new words and to submit a new request to the index. This supplementary step might give to LDA an advantage. Some documents overlooked by a normal keyword search can appear very relevant once the results are broken down into topics. However, LDA does not provide a ranking of topics nor of documents, which costs our system in µ-ERR performance and let the combination of techniques have a better balance between the two metrics with only a little loss over the CubeTest metric.

The two last rows of Table 1 for the main task show systems that use more than two iterations of results (more than one feedback from user). Ul_lda.roc.3 uses three iterations and Ul_lda.roc.10 uses two to ten iterations to explore results. As we can see, this lengthier exploration has been strongly penalized by the Cube Test metric. Since many of the later iterations were "false starts", the metric penalized them as a waste of user time.

For the "judged-only" task, we submitted only one run of our algorithm to complete the limit of five runs. Our experiments indicated previously that the combination of algorithms is the best system from the point of view of the Cube Test metric. Results indicate that it performs well compared to other systems from this point of view too. Indeed, this system provides the best results for nearly 40% of the topics and has a result over the median of the group of participants for 98.3% of the topics.

VIII. Conclusion

In this paper, we presented the user-feedback-based iterative search system we implemented to solve the TREC Dynamic Domain track. We evaluated different algorithms to try to diversify our results and cover a maximum of interesting topics. We also tried different strategies to re-orient our search based on user feedback to. During our experiments, we had to cope with various difficulties such as the important amount of irrelevant or un-judged documents, the difficulty to re-orient the search after a "false-start" situation, and the importance of quickly finding results to avoid important time penalty. The final combination of techniques we selected provides a system capable of balancing exploration and exploitation. Our experimental results indicate that this system performed as the best system for both CubeTest and µ-ERR.

In future work, and possibly for TREC 2016, we plan to handle the "false-start" by using active learning to train a classifier that could predict if a document will be relevant or not. We will also try a different basic search engine instead of Solr, to see how it impacts the results.

IX. References

Andrzejewski, D., & Buttler, D. (2011). *Latent topic feedback for information retrieval.* Paper presented at the Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining.

Chapelle, O., Metlzer, D., Zhang, Y., & Grinspan, P. (2009). *Expected reciprocal rank for graded relevance.* Paper presented at the Proceedings of the 18th ACM conference on Information and knowledge management.

Cohn, D., Caruana, R., & McCallum, A. (2003). Semi-supervised clustering with user feedback. *Constrained Clustering: Advances in Algorithms, Theory, and Applications, 4*(1), 17-32.

Luo, J., Wing, C., Yang, H., & Hearst, M. (2013). *The water filling model and the cube test: multi-dimensional evaluation for professional search.* Paper presented at the Proceedings of the 22nd ACM international conference on Conference on information & knowledge management.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval* (Vol. 1): Cambridge university press Cambridge.

Salton, G., & Buckley, C. (1997). Improving retrieval performance by relevance feedback. *Readings in information retrieval, 24*(5), 355-363.

Steinbach, M., Karypis, G., & Kumar, V. (2000). *A comparison of document clustering techniques.* Paper presented at the KDD workshop on text mining.

Yang, H., Frank, J. & Soboroff I. (2016). *TREC 2015 Dynamic Domain Track Overview*