A User-in-the-Loop Process for Investigational Search: Foreseer in TREC 2013 Microblog Track

Cheng Li, Yue Wang, Qiaozhu Mei
School of Information, Department of EECS, School of Information
University of Michigan
{lichengz, raywang, qmei}@umich.edu

ABSTRACT

Traditionally, ad hoc retrieval aims at satisfying an information need with a few highly relevant documents. This *high-precision* approach works well for simple and clear queries. When the information need becomes complex, a few top-ranked documents may not provide satisfactory answer. As a result, the user adapts to reformulate the query to explore more relevant information; the search engine evolves to diversify results to cover the user's real information need. In cases where the user puts emphasis on *high recall* of the results, the search process becomes increasingly laborious.

We propose a retrieval system that interacts with the user to obtain high precision and high recall search results, while minimizing the user effort. It iteratively explores the collection by a series of queries to optimize the recall, and refines an active learning classifier to maintain the precision. We built a prototype of the system for TREC 2013 Microblog Track. Depending on the actual query, the system converges to a stable decision on relevant/non-relevant tweets after asking for a few hundred labels, which was used to retrieve and rank 10,000 tweets (maximum allowance of the TREC API).

1. INTRODUCTION

Traditionally, web searches are grouped into three categories: informational, navigational and transactional [1]. Users with informational search queries explore various information covering a broad topic (e.g., football). Navigational search aims at the official website or home page of a single entity (e.g., youtube). Transactional search queries are issued when users plan to perform an action on the web, like ordering a book and downloading a software.

The three categories seem to cover user intentions in most cases. However, there is no emphasis placed on these search actions where the users wants a *high*, or even a *full* coverage of relevant documents pertaining to a query, while still keeping a good precision. We call such task investigational search, as opposed to the three existing search classes.

There are cases when users requires more than just top-ranked documents and investigational search is more desirable, among which is the microblog search. Searching tweets to satisfy one's information need can be a tricky task. Consider the query "buying clothes online", where the searcher might look for online stores, coupons, discussions, suggestions, or even remote try-on technologies that are related to online apparel shopping. Since each tweet only contains very brief comment from an individual perspective, a few top-ranked tweets often deliver incomplete information and biased opinion. It is almost inevitable for the searcher to reformulate the query and search multiple times, sift through voluminous redundant and less relevant tweets, and finally put discrete pieces together into a holistic story. The process becomes increasingly laborious as the searcher proceeds to look for yet another tweet with fresh information.

There are other scenarios when information need cannot be satisfied by only a few top-ranked documents, not limited to microblog search. The challenges arise for several reasons:

Poor single-document coverage The documents are so short (or biased) that none of which independently provides useful information. Shards of texts, such as microblogs, are typical examples.

Elusive information need As the user realizes new relevant contents in top-ranked documents, he internally refines the definition of "relevance" and reformulates the query. In other words, the mature definition of information need may not exist during the composition of the initial query, but takes shape in the process of interacting with the search engine. This is evidenced by "sessions" in search logs.

Mismatched search goals Ad hoc retrieval assumes emphasis on *high precision*, but the user places emphasis on *high recall*. In other words, missing any relevant document imposes high risk. Such situations include patent search, legal search/e-discovery, scientific literature review, medical record search, etc.

Various efforts are made by IR researchers and practitioners to tackle these challenges. Query suggestion generates multiple candidate queries to help the user clarify the information need [3]. Diversified search aims to return a ranked list of

Text Retrieval Conference TREC 2013 Microblog track

documents that complement each other to provide complete coverage for a query [4]. However, due to the *risk-averse* nature of traditional search engines, they tend to conservatively return highly relevant results at the top, rather than probe (and agitate) the user with border cases of relevant vs. non-relevant documents. Therefore it is not ideal to use a search engine to explore the *complete set* of relevant information pertaining to a query.

By contrast, active learning is *risk-seeking* by design. A binary active learning classifier will proactively ask for clarification on uncertain/border cases, learning a decision boundary to separate the dataset into positive and negative classes [5]. Unfortunately, active learning can be ineffective when the class distribution is both extremely imbalanced and unknown *a priori*, which is common in IR (number of relevant vs. non-relevant documents for a query). Also, it is not typical settings for active learning when the dataset is too large to allow full access, but only allow limited access every time (e.g. via search).

In response to these challenges, we propose a retrieval system that combines ad hoc retrieval and active learning in a unified process. It interacts with the user to maximize the opportunity of obtaining as complete as possible set of documents satisfying an information need, while minimizing the user's efforts. The system alternates between *inductive* and *deductive* stages, emulating the iterative process of knowledge development. In the inductive stage, it refines information need by active learning; in the deductive stage, it converts current knowledge into a new query and searches the collection for more relevant documents.

Our user-in-the-loop system is designed for investigational purpose, where *high recall* is prioritized. The user is an "investigator" who would like to invest efforts to obtain coverage of relevant documents as complete as possible. Examples of such users may include legal professionals performing e-discovery tasks, patent agents searching for overlapping inventions, researchers surveying related work, physicians studying possible outcomes of a treatment, or even fans digging into every single piece of gossip about a celebrity.

The rest of the paper is organized as follows. Section 2 gives an overview of the system architecture and each component. Section 3 describes the implementation of the prototype that we built for Microblog Track, which is an instantiation of the system. Section 4 reports the evaluation and discusses about the results. Finally, we conclude the paper in Section 5.

2. SYSTEM ARCHITECTURE

In this section, we give a high-level overview of the system architecture and each component.

2.1 Overview

The system architecture is shown in Figure 1. At the very beginning, the investigator comes up with an initial query. After initial retrieval, a binary classifier will proactively select a few documents and ask the oracle (e.g. human investigator) to label them as "relevant" or "nonrelevant". The active learning goes on until the classifier's prediction performance is reasonably good. The system will then prepare a new query based on the learned model, the original query, and other heuristics; if necessary, the investigator can also intervene and modify the query. The new query will retrieve another set of documents, which is merged into the retrieved document pool. At this point, a second round of active learning starts to classify the updated pool. The process continues until the binary classifier performs reasonably well without asking for labels.

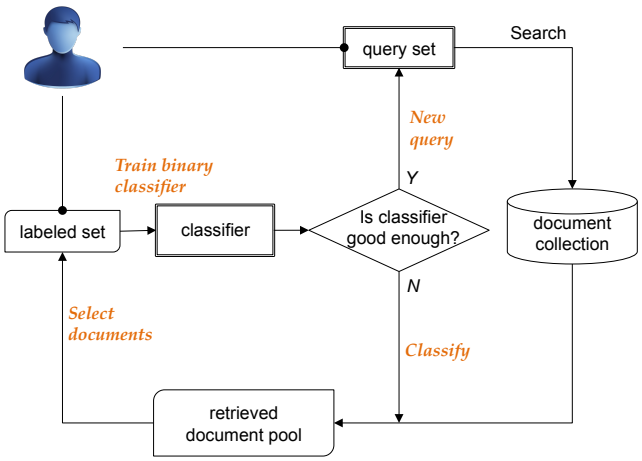

Figure 1: System architecture.

2.2 Search

This component performs ad hoc retrieval: given a query, the retrieval function evaluates documents in the collection and returns a ranked list of documents ordered by relevance. The retrieval function can take any reasonable implementation, such as Boolean retrieval model, vector space model (e.g. BM25), language model (e.g. Dirichlet prior) [2]. Many public search services, such as Google Search API[1] and Yahoo! BOSS API[2], impose rate limits on the number of documents returned by one query. As in TREC 2013 Microblog Track, the ranking function is Dirichlet prior smoothing language modeling provided by TREC API, with maximum number of retrieve 10,000.

Each time the retrieved documents will be merged into the current retrieved document pool, which is the dataset for active learning.

2.3 Active Learning

The task of this component is to learn a binary classifier to separate relevant documents from non-relevant ones, in the currently retrieved document pool. Normally, a search engine can only passively receive explicit or implicit relevance feedback from clickthrough data and estimate a relevance model. In contrast, our system uses active feedback: the active learning component proactively asks the investigator to evaluate the relevance of documents, which may include non-relevant/border cases. The promise of active learning is that by allowing the algorithm to select data to label, it will perform better with less training. Here it learns a

[1] `https://developers.google.com/custom-search/json-api/v1/overview`
[2] `http://developer.yahoo.com/boss/search/`

good model of relevance with less labeling effort from the investigator.

When newly retrieved documents arrive at the unlabeled document pool, selecting documents needs special care. In the starting round, the system has *no* labeled documents, yet must select a subset from the unlabeled pool and ask for labels. Moreover, it is desirable to include both positive and negative examples in the subset for initial training. This is known as the *cold start* problem commonly seen in data modeling tasks. It is legitimate to assume that top-ranked documents are likely to be relevant, while low-ranked documents are more likely to be non-relevant. Therefore, one possible strategy is to select several top ranked documents as well as cluster centroids of the the rest documents, and ask for labeling.

In subsequent retrievals, even though the system has accumulated some labeled examples and trained an inaccurate classifier, newly retrieved documents are still unseen to the classifier. If many new and similar documents fall into the uncertain region of the classifier, active learning tends to ask labels for all these documents. To accelerate the learning rate and reduce the labeling efforts, a sensible strategy is to select random samples or cluster centroids of these uncertain documents for labeling.

At the end of each inductive stage, we have a binary classifier that automatically separates the unlabeled documents into relevant and non-relevant classes.

2.4 Query Reformulation

When the binary classifier is perceived to be performing reasonably well, the investigator may want to proceed and retrieve another set of documents for inspection. Undoubtedly, he hopes that the newly retrieved documents are both relevant and different. To achieve this goal, a new query has to be carefully formulated.

The set of feedback documents and classifier-judged documents from the previous inductive stage can be exploited to update the query. Feedback-based query expansion is known to be effective in improving recall while maintaining precision. This component may also incorporate other query expansion strategies, such as knowledge-based query expansion. Furthermore, the investigator himself may intervene and edit the query directly.

Clearly there is exploration-exploitation trade-off in query reformulation component. Too much exploration will retrieve many non-relevant documents, increasing the burden of labeling; too much exploitation will limit the system from getting broad coverage. A good query reformulation strategy should balance these two.

2.5 Stopping Criteria

The system stops whenever any of the following conditions occurs:

- The retrieval function cannot pick up anymore unseen relevant document;
- The binary classifier has become robust enough to label any unseen document;
- Labeling budget is used up.

After the system stops, the relevant document set can be identified by applying the binary classifier on all the retrieved data. It should be noted that even in the first two cases, it is not guaranteed that the system has identified the "true complete set" of relevant documents. The coverage depends on the initial query, the design of the retrieval function, selecting strategies of active learning, and query reformulation strategy.

3. IMPLEMENTATION

In this section, we describe the prototype implementation of the system for TREC 2013 Microblog Track.

The pseudo code for the prototype is displayed in Algorithm 1. As we can see, this is a very general process. Many suitable classifier can be used in the active learning process, or the inner loop. A variety of query expansion techniques could be integrated into the flow as well. This system is able to be easily adapted to any other data collection tasks as long as the API is provided. We will elaborate each step of the retrieval process in next few subsections.

Algorithm 1 User-in-the-loop process

Input: Initial query q_0
Output: A set of relevant tweets

loop
  $RetriT \leftarrow$ tweets collected through TREC API using query q_i
  $UnivT \leftarrow UnivT + RetriT$
  if $i = 0$ **then**
    Do clustering on $RetriT$
    $CentT \leftarrow$ tweets closest to clustering centroids
    $UncertainT \leftarrow CentT+$ top 10 tweets of $RetriT$
  end if
  repeat
    Request human labels for $UncertainT$
    $LabeledT \leftarrow LabeledT + UncertainT$
    $UnlabeledT \leftarrow UnivT - LabeledT$
    Classifier predicts labels for $UnlabeledT$
    $UncertainT$ is updated by classifier
  until Classifier performs well
  $q_{i+1} \leftarrow q_0+$ query expansion based on current model
end loop

3.1 Text Preprocessing

Text are preprocessed for later usage. Specifically, tweets starting with "RT" and non-English tweets are removed. Near duplicates are also removed to reduce the burden on humans and algorithms. These duplicates are added back at last if the tweet they resemble are predicted as relevant.

3.2 Clustering

Clustering is only performed after the very first outer loop. This is used to solve the cold start problem, where the classifier has no training data to build model and thus do prediction. Sampling tweets randomly for humans to label could

be another easier option. To increase the diversity of tweets being selected, we do K-means clustering, and pick the cluster centroid tweets. Top tweets retrieved from TREC API are also included, with the assumption that they are mostly positive tweets. Including them could reduce the possibility that initial training data set is dominated by negative tweets.

3.3 Classification

We choose SVM with linear kernel as our classifier. Krovetz stemming is performed on the text and stopwords are kept because we found that many features with stopwords included are capable of distinguishing relevant documents from irrelevant ones. The features we used are unigrams, bigrams, and trigrams. Only most frequent trigrams are kept to avoid having a high dimensionality of feature space. After training, SVM will return a model, specifying the weights of each feature. We use the linear combination of weights to compute the confidence score of the unlabeled tweets. Among these tweets, 10 positive and negative tweets with least absolute confidence values are chosen, waiting for human to label.

3.4 Human Labeling

We build a web interface to facilitate human annotation. When displaying the tweets, important positive and negative features extracted by SVM are highlighted using different colors. Highlighting reduces user's process time of one tweet. However, admittedly, this could be misleading and chances are increased that unhighlighted but important words might slip away.

In addition to uncertain tweets, a random sample of confident tweets are also presented in the webpage. This helps users to monitor the progress of the classifier. When the classifier is doing well on both uncertain tweets and confident tweets, it's usually a good sign to stop the current inner loop and proceed to the next outer loop, namely, using the expanded query to retrieve more tweets through TREC API.

Expanded query, updated per each inner loop, is also shown on the interface. Users are allowed to modify the query, which is useful for domain experts.

3.5 Query Expansion

We use important positive features as candidates for query expansion. Since TREC API supports weighting of terms, each feature's weight provided by SVM model is directly used as the weight for query. In the future work, we should use external knowledge, such as knowledge base as a source of query expansion.

Note that the initial query is always kept in later queries. This approach alleviates the problem of semantic drift.

3.6 Reranking Tweets in Submitted Runs

Since we have cast the ranking problem as a task of classification, we need to rerank the tweets for submissions. There are two quick ways to obtain a ranked list of tweets.

1. Rank tweets by the confidence score using the weights of features given by SVM
2. Rank tweets by Dirichlet model, which is already implemented by the TREC API.

With the two ranking methods at hand, the next thing is to combine them to form ranking lists. To this end, we divide the entire set of retrieved tweets into four parts.

- Part (a): Labeled positive tweets
- Part (b): Labeled negative tweets
- Part (c): Unlabeled tweets predicted as positive by SVM
- Part (d): Unlabeled tweets predicted as negative by SVM

Tweets from part (b) are discarded for all the submissions. Note that in the four submissions, the rank of tweets are calculated separately for part (a), (c), and (d). This is to ensure that tweets from part (a) are always ranked higher than tweets from part (c) , which are in turn ranked higher than part (d).

The ways we form our four ranking lists are listed as follows:

1. **FSsvm**: Rank tweets in (a), (c), and (d) by SVM respectively, and concatenate them.
2. **Direrank**: Labeled positive tweets ranked by Dirichlet model, concatenated by unlabeled tweets ranked by Dirichlet model. In particular, we submit to the TREC API the final query generated by SVM and get a ranked list of tweets returned by API. If the tweets retrieved in the process are in the returned list, rank them according to the returned list. If they are not, rank them according to (1), and append them to the returned list.
3. **Avgrank**: Average rank of SVM and Dirichlet model. Tweet ranking is decided by $rank_3 = rank_1 + rank_2$.
4. **RvsDir**: Tweets that are ranked high by SVM, but low by Dirichlet model. This is used to guarantee that hard tweets found by the system could be judged by TREC raters. For part (a) and (b), rank is decided by the formula $rank_4 = rank_1 - rank_2$. For part (c), the formula is $rank_4 = rank_1 + rank_2$. Tweets with smaller rank scores are ranked higher, even though the score could be negative.

4. EXPERIMENTS AND DISCUSSIONS

Performance evaluation is shown in Table 1. To illustrate the set relations of NIST qrels, our labeled set, and the 4 submitted runs, Venn diagrams are drawn in Figure 2 - 6.

In total, 32,237 tweets were labeled for 60 topics, half the number of tweets (qrels) evaluated by TREC assessors. It means that on average 537 tweets were labeled to train a binary classifier for each topic, a non-trivial amount of human efforts. We argue that user-in-the-loop retrieval system targets at users who are willing to invest substantial time and efforts in the search task.

In Figure 2, Set G and H are disputes between NIST qrels and our labels. A second inspection reveals that most NIST-judged relevant tweets are indeed relevant, while many NIST-judged non-relevant tweets might also be justified as relevant. It means that our approach missed some relevant tweets in Set A. Future work will improve the components for a higher recall. It's also worth noting that a good proportion of tweets labeled as positive by our system is not included in the NIST evaluation pool at all. This implies the inadequacy of the "cut-off-at-K" pooling strategy used in the TREC evaluation when *high-recall* is emphasized.

SVM classifiers tend to draw the decision boundary conservatively, especially when positive training examples are few. Therefore, it maintains good precision at the retrieved set level (see Figure 5. However, pure SVM scores $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ do not rank high-quality tweets to the top, since tweets with extremely high score tend to be overfitting cases. Such tweets merely repeat words with positive weights. Dirichlet prior smooths the document language model, promoting documents with diverse on-topic words, penalizing those overfitting tweets. However, the smoothed language model may also promote non-relevant (false positive) tweets, which will hurt precision. By averaging the ranks of SVM and Dirichlet model, we obtain improved precision at the top (Table 1, Avgrank).

	R-prec	MAP	P@30
FSsvm	0.4882	0.4413	0.6744
Direrank	0.5172	0.4735	0.6967
Avgrank	0.5033	0.4570	0.7061
RvsDir	0.4931	0.4010	0.5822
Auto.: average best	0.5214	0.4820	0.7222
Auto.: average median	0.2721	0.2126	0.4217
All: average best	0.6086	0.5737	0.7989
All: average median	0.2834	0.2212	0.4311

Table 1: Performance evaluation. Auto.: 65 automatic runs; All: all runs (65 automatic + 6 manual); average best: average of best performance per topic; average median: average of median performance per topic

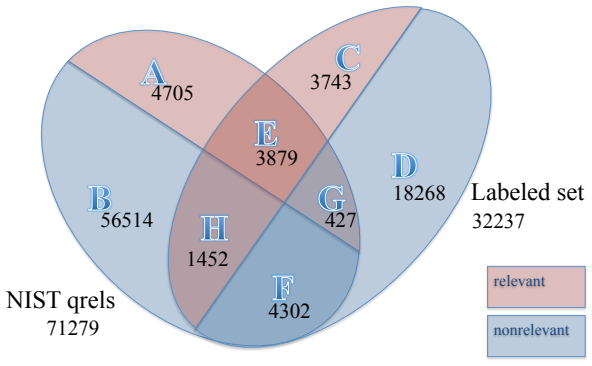

Figure 2: Venn diagram of NIST qrels and our labeled set. Numbers are sizes of corresponding sets.

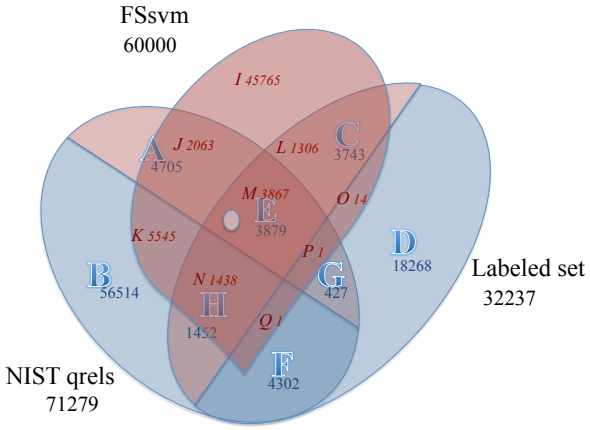

Figure 3: Venn diagram of NIST qrels, our labeled set, and FSsvm. Numbers are sizes of corresponding sets.

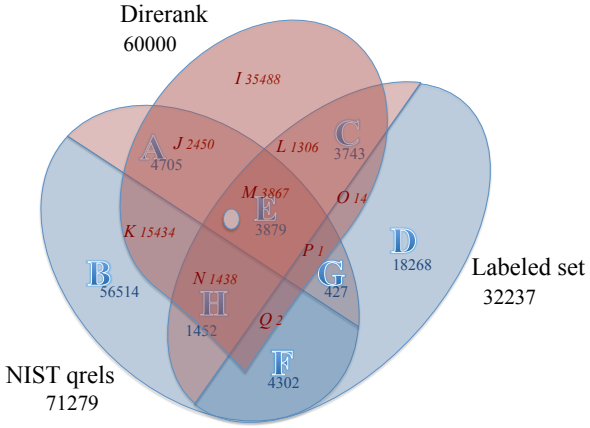

Figure 4: Venn diagram of NIST qrels, our labeled set, and Direrank. Numbers are sizes of corresponding sets.

5. CONCLUSIONS

In this study, we proposed a retrieval system for investigational purposes, where *high recall* is prioritized. It is justified by many use cases, including e-discovery, literature review, medical record search, etc. The system brings ad hoc retrieval and active learning in a unified process, aiming to reduce the user's effort in pulling out as many relevant documents as possible.We built a prototype to demonstrate the system in TREC 2013 Microblog Track. Various parts of the system can be improved in future work, so that with less labeling effort, the system still maintains or even improves on both precision and recall.

ACKNOWLEDGMENT

This work is partially supported by the National Science Foundation under grant numbers IIS-0968489, IIS-1054199, and CCF-1048168, and partially supported by the DARPA under award number W911NF-12-1-0037. We thank Paul Resnick and Samuel Carton for their helpful suggestions in this work.

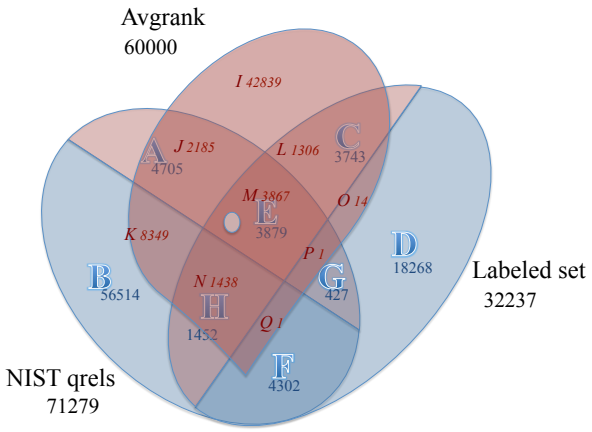

Figure 5: Venn diagram of NIST qrels, our labeled set, and Avgrank. Numbers are sizes of corresponding sets.

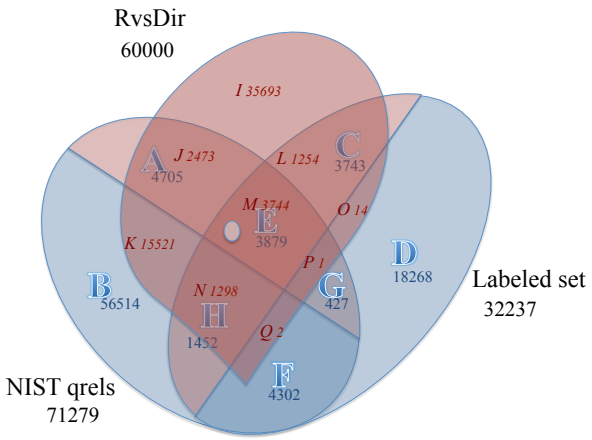

Figure 6: Venn diagram of NIST qrels, our labeled set, and Rvsrank. Numbers are sizes of corresponding sets.

6. REFERENCES

[1] A. Broder. A taxonomy of web search. In *ACM SIGIR forum*, volume 36, pages 3–10. ACM, 2002.

[2] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[3] Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 469–478. ACM, 2008.

[4] K. Raman, P. N. Bennett, and K. Collins-Thompson. Toward whole-session relevance: Exploring intrinsic diversity in web search. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 463–472. ACM, 2013.

[5] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.