USING DOCUMENT SIMILARITY NETWORKS TO EVALUATE RETRIEVAL SYSTEMS

by

Aparna Kailasam

An abstract of a thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

May 2010

Approved: _____

Ben Carterette, Ph.D.
Professor in charge of thesis

USING DOCUMENT SIMILARITY NETWORKS TO EVALUATE RETRIEVAL SYSTEMS

by

Aparna Kailasam

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

May 2010

USING DOCUMENT SIMILARITY NETWORKS TO EVALUATE RETRIEVAL SYSTEMS

by

Aparna Kailasam

Approved: _____

Ben Carterette, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee
Department of Computer and Information Sciences

Approved: _____

B. David Saunders, Ph.D.
Chair of the Department of Computer and Information Sciences

Approved: _____

George H. Watson, Ph.D.
Dean of the College of Arts and Sciences

Approved: _____

Debra Hess Norris, M.S.
Vice Provost for Graduate and Professional Education

Acknowledgments

I would like to take this opportunity to express my sincere gratitude to my advisor, Professor Ben Carterette, for his constant support and encouragement throughout the course of this thesis work.

My interest in Information Retrieval began when I took a course in the subject matter taught by Professor Carterette in Spring 2009. His enthusiasm and professionalism in teaching the course inspired me to work harder and prompted me to request that he grant me an opportunity to work on a research project under him in the following summer. It was the glimpse of hands-on research experience I had during this period that convinced me to do a thesis under his guidance. Professor Carterette has been an excellent mentor and a constant source of invaluable ideas.

I would also like to thank all my friends at the University of Delaware for the fantastic and fun-filled two years I spent there. I would especially like to thank Shailaja Rabindran and her family for providing me with constant and unconditional support during my stay in their home in Delaware. I am also grateful to her for letting me share the love of her two beautiful and adorable pet cats, Dusty and Raja.

Though my family is located in India, they never felt far away as they always reached out to me whenever I needed them. I cannot thank my parents or my sister Anupama enough for being my pillars of support throughout.

Table of Contents

List of Figures

List of Tables

Abstract

Information Retrieval System Evaluation is important in order to determine the performance of a system's ability in satisfying the information needs of users. Evaluation of retrieval systems requires constructing test collections, which consists of obtaining documents, topics (information needs) and a set of relevance judgments that indicate the relevant documents for a topic. For a small collection of documents, it is possible to judge every document for relevance. A large document collection will demand an enormous judging effort, which is unrealistic.

We present a novel technique for evaluating retrieval systems using minimal human judgments. Our proposed solution initially selects a small 'seed-set' of documents which are judged for relevance. The relevance information from this set is then propagated through an adhoc network of document similarities to determine the relevance of other unjudged documents. The original judgments combined with the inferred judgments constitute the complete set of judgments that is used to compare the relative performance of different systems. Our results show that we can effectively compare different retrieval systems using very few relevance judgments and at the same time achieve a high correlation with the true rankings of systems.

Chapter 1

INTRODUCTION

The goal of Information Retrieval is to help users find arbitrary information in very large collections of unstructured text or other media. One could be searching the World Wide Web or looking for scholarly articles. Typically, retrieval systems accept queries from users. The system uses some representation of the text in the query and text in documents to calculate a matching score between the query and each document, then returns a ranked list of documents in decreasing order of score to the user.

IR systems are complicated. There are many components that interact with each other to influence the final results, including the text representation, the scoring function, properties of the collection, and so on. Researchers and developers of IR systems need to know whether they can expect users of their systems to be happy with the results provided, but because the system is so complicated, there is no way they would be able to predict how any given design decision will ultimately affect the users. It is therefore necessary to evaluate the final ranked list itself. This is done by evaluating the *relevance* of the documents in the collection.

Information Retrieval System Evaluation follows the so-called Cranfield paradigm developed in the 1960's by library scientists at Cranfield Aeronautics. The Cranfield paradigm uses a static test collection for batch evaluation over a representative sample of queries. A test collection consists of documents, a set of queries and a fixed set of relevance judgments. Documents solve information need, queries are information needs used to search against the documents, and the relevance judgments identify the relevant documents for the topics. The Cranfield paradigm assumes that the relevance judgments are complete; every document is

judged with respect to every topic. If the document collection size is huge, this task requires exhaustive human effort, and in reality becomes infeasible and inefficient.

The Text REtrieval Conference (TREC) was started by researchers at NIST in 1992 as a way to involve the information retrieval research and development community in the construction of large-scale test collections for research [27]. Before TREC, test collections for IR research consisted of a few thousand short document abstracts; they had to be small because of the cost involved in getting human relevance judgments. TREC scaled up to hundreds of thousands full-text news articles. It did this by use of the *pooling method* [22]. First, a set of queries with full statements of an information need were developed at NIST. These queries were sent to research groups at universities and companies that had agreed to participate. Each group submitted the queries to one or more retrieval systems that had indexed the news articles. They returned their results to NIST, which then pooled the top 100 documents ranked by each system for each query. Every document in the pool was judged for relevance. Although only a small fraction of the total collection was judged, since they were documents that were actually retrieved by modern systems, it was good enough to provide accurate evaluation.

Since 1992, the document collections used for TREC have grown. In 2004 a 25-million document collection was introduced [11]. 2009 saw the introduction of a one *billion* document collection [12]. As collection sizes increase, the amount of human effort needed even for shallow-depth pools increases. The cost is beyond what NIST can expend. Furthermore, research groups that want to work on new problems outside of TREC cannot rely on NIST to obtain relevance judgments. For these reasons it is important to have ways to build test collections cheaply while still allowing accurate comparisons between retrieval algorithms.

Our goal in this work is to propose a semi-automatic evaluation system that infers the relevance of unjudged documents given minimal judgments and use this information to evaluate different systems. Our approach starts by selecting a seed-set of documents that have been judged for relevance. We construct an adhoc network of document similarities that consist of both judged and unjudged documents and propagate the relevance information

from the seed-set to unjudged documents in the network. We then use the inferred relevance judgments to evaluate retrieval systems.

Our approach is applicable for use in the following environments that follow the Cranfield-based evaluation strategy: Text REtreival Conference (TREC), NII-NACSIS Test collection for IR Systems (NTCIR), Cross-Language Evaluation Forum (CLEF), INitiative for the Evaluation of XML Retrieval (INEX). Another scenario is a research group that wants to evaluate their retrieval system but does not have access to sufficient resources for obtaining relevance judgments for their collection of documents. They could start with few judgments and infer the relevance of unjudged documents and use this to evaluate the performance of their retrieval system.

We will show that our method can predict the relevance of unjudged documents with 62% precision and 80% recall and that bears a Kendall τ correlation of 0.95 with the original ranking. Despite the relatively low success rate, we can compare different systems to each other with high accuracy, that is evident from the high value for τ correlation: we will show that we can use a small seed set of human judgments to rank a set of systems in increasing order of quality very close to the ranking that would be obtained with a much larger set.

Our main contributions include the following:

- We present techniques to estimate relevance of an unjuded document given few relevance judgments to begin with
- We propose a technique for retrieval system evaluation using a network algorithm

The rest of the report is organized as follows: Chapter 2 gives a background of prior related work in evaluation of retrieval systems and popular link-graph based approaches for ranking documents. Chapter 3 provides an overview of and a detailed description of our algorithm. Chapter 4 presents an overview of our experimental procedure. Experimental results are presented in Chapter 5. We finally conclude the report in Chapter 6.

Chapter 2

PREVIOUS WORK

As described before, our goal is to evaluate retrieval systems using a combination of minimal human relevance judgments and inferred judgments based on similarity networks. This work, therefore, combines research on evaluation with incomplete and inferred judgments with research on propagation of information through networks. In this section, we summarize previous work in these two areas. First we will introduce the standard measures used to evaluate retrieval systems.

2.1 IR Evaluation Measures

As we discussed in Chapter 1, information retrieval evaluation is based on human judgments of relevance to individual documents. It is useful to be able to distill these judgments down to a single measure or set of measures of properties of a ranked list of documents. The information retrieval literature contains a wide variety of such measures; a handful of these are widely used by researchers and developers.

Precision is the proportion of documents retrieved that are actually relevant according to human assessors. *Recall* is the proportion of relevant documents in the entire collection that have been retrieved. Precision and recall are usually calculated at a particular rank cut-off, for instance precision at rank 10 is the proportion of the top 10 ranked documents that are relevant, and recall at rank 10 is the proportion of relevant documents in the collection that have been ranked in the top 10. Precision and recall are frequently combined into a single summary measure. F is the harmonic mean of precision and recall at a particular

rank. R-precision (or breakeven point) is the precision at the rank equal to the number of relevant documents.

Precision may also be averaged over the entire ranked list for a summary of performance. The average of the precisions computed at each rank that a relevant document has been retrieved is called *average precision* (AP). AP is equivalent to the area under a curve of recall versus precision.

Because different queries can have very different performance, it is standard to evaluate a set of queries, computing AP or some other measure for every query in the set. These measures are then averaged over the set, and frequently a confidence interval calculated for comparison to other systems. When AP is averaged over a set of queries, it is called *mean average precision* (MAP). MAP is the most common measure seen in IR literature today. It is the official measure of the TREC conference, and it is the measure we will use throughout this work.

The measures above are based on binary relevance judgments. Relevance could also be judged on a graded scale (highly relevant, relevant, not relevant) for finer distinctions between documents. *Discounted cumulative gain* (DCG) is an evaluation measure that makes use of graded relevance judgments. For each rank, it calculates a 'gain' as the relevance grade of the document at that rank divided by a logarithmic discount. These gains are summed over ranks to produce the final cumulative measure of performance. DCG and the normalized variant NDCG have become standard evaluation measures for search engines like Google.

2.2 Low-Cost Evaluation

Work on low-cost evaluation can be seen as falling along three dimensions:

1. Determining which documents to judge in order to provide efficient and reliable evaluation results

2. Using a small, incomplete set of judgments to reliably evaluate systems

3. Inferring the relevance of unjudged documents in order to produce better estimates of evaluation measures when judgments are missing

These three dimensions are not independent of one another, of course; a certain estimate of evaluation measures based on incomplete judgments may require a certain approach to collecting those judgments. Nevertheless, we find this a useful classification scheme.

2.2.1 Choosing Documents to Judge

Large-scale retrieval evaluation requires efficient construction of large test collections. A test collection consists of 3 components, namely:

1. Documents
2. Queries
3. Relevance judgments

If the document collection size is huge, it may become infeasible to judge each and every document with respect to a topic. Hence, it might be good to judge only a *subset* of documents to evaluate different systems; given that these documents are well-representative of the entire set of relevant documents. The following approaches have been proposed:

The Pooling [22] method involves forming a pool by taking the union of the top k documents retrieved by n independent systems for a query. Assessors judge every document in the pool for relevance to that query. It is justified by the principle that if a retrieval system is effective, the top-ranked documents will be excellent candidates for inclusion in the subset of documents to be judged. If k and n are large, the set of judged relevant documents is assumed to be representative of the entire set of relevant documents, and hence suitable to evaluate different systems. But, on the other hand, the number of documents to be judged for a topic (up to kn) is still huge. Also, the subset of documents closely approximates the ideal set of relevant documents only under the assumption that all systems achieve equal retrieval effectiveness, which is not realistic.

Cormack et al. proposed 2 methods [14]; the first, called *Interactive Searching and Judging (ISJ)*, combines the efforts of multiple searchers in identifying a representative set of the set of relevant documents. This method ranks each document based on the length and the number of passages satisfying the query. The assessors are not stuck with a pool, they are able to submit queries themselves and judge the documents that come back. They're allowed to do this iteratively so that they can learn about the topic and find more relevant documents. Even though this method uses roughly one-quarters of a judging effort compared to Pooling, it has been shown to be effective in comparing retrieval performance. The second, called *Move to Front Pooling (MTF)*, selects the next document to be judged from the system that retrieved the most recently judged relevant document. MTF can be seen as ordering documents in a priority queue by a rough approximation of their probability of relevance, adaptively updating the probabilities after each judgment. This method finds more relevant documents more efficiently than the traditional pooling method.

Carterette et al. proposed the *Minimal Test Collections* method [10] that selects those documents for judging that are the most informative about the difference in the MAP of two retrieval systems. It is an iterative algorithm that assigns weights to documents based on the judgments that were made up to that point and these weights are recalculated after each judgment is made. hey define a sampling prior based on cumulative information about the ranks at which documents appear in the ranked lists being sampled from. This method is capable of ranking systems with a high degree of confidence using minimal judgments and is also applicable for use in non-TREC research environments.

The method by Aslam et al. in [4] selects documents for judging from a specific random sample of documents from the given ranked lists to produce unbiased, low-variance estimates of retrieval measures like average precision. This method can also take information coming from additional judged documents from other, non-random locations in a ranked list.

2.2.2 Evaluation with Incomplete Judgments

The simplest approach to estimate evaluation performance when judgments are incomplete is to assume that all unjudged documents are non-relevant. This assumption is reasonable when the relevance judgments are fairly substantial; but if the document collection is very large, it is likely that many relevant documents exist beyond those that have been judged. Also, performance can be better estimated by taking unjudged documents into account. Therefore, other estimation methods become necessary.

Buckley & Voorhees proposed the *bpref* evaluation measure [7] that is calculated only over judged relevant and nonrelevant documents, ignoring unjudged documents *bpref* computes a preference relation that is a function of the number of times judged non-relevant documents are retrieved and ranked higher than judged relevant documents. bpref has been shown to rank systems similar to average precision, and is known to be highly stable given incomplete relevance judgments

Yilmaz & Aslam's propose *inferred average precision* [28] is an estimate of average precision on a set of judgments uniformly sampled from the original judgments set. In effect, this set acts as the set of incomplete judgments. The notion of *inferred average precision* arises from thinking about average precision as the outcome of a random experiment. A relevant document is chosen at random from a ranked list of documents and the probability of finding a relevant document at or above that rank is determined; that in effect corresponds to finding the precision at that rank. Randomly picking a relevant document corresponds to averaging these precisions over all relevant documents.

Carterette et al's *Expected Average Precision(EAP)* [10] is calculated as an expected value of average precision expressed as a function of document relevance variables. The relevance of each document is represented as a Boolean random variable X_i, and the probability that the document is relevant is $p(X_i = 1)$. For a judged relevant document $p(X_i = 1) = 1$ and for a judged nonrelevant document $p(X_i = 1) = 0$. Consequently, any evaluation measure (and thus AP) can be expressed in terms of document relevances X_i, and thus must have a distribution over possible assignments of relevance to unjudged documents. The maximum

a posteriori ranking of systems is one in which systems are ranked by the *expected* value of average precision. Given probabilities of relevance $p_i = p(X_i)$:

$$EAP \approx \frac{1}{\sum p_i} \sum_{i}^{n} c_{ii} p_i + \sum c_{ij} p_i p_j$$

Also, since Mean Average Precision is the average of the average precisions computed for a set of topics T, EMAP can be computed as:

$$EMAP = \frac{1}{T} \sum_{t \in T} E[AP_t] \tag{2.1}$$

2.2.3 Inferring Relevance

Instead of trying to work around missing judgments, we can use evidence provided by any relevance judgments we have to infer the relevance of unjudged documents. These inferred judgments can be used to augment the existing judgments, possibly providing more accurate evaluation.

Carterette et al. present a model of evaluation (described above) that can incorporate any type of evidence to estimate relevance [9]. The similarity of an unjudged document to other judged relevant documents may provide evidence for that document being relevant; this idea comes from the *cluster hypothesis*, which says that associated documents tend to be relevant to the same requests. By modelling the relevance of a document conditional on its similarity to judged relevant documents, the relevance of unjudged documents is estimated. This method is capable of ranking retrieval systems with minimal human effort compared to traditional TREC methods like pooling.

Aslam et al. present a technique that infers the relevance of unjudged documents by obtaining estimates of average precision and the number of relevant documents [4] by sampling and judging some documents. Based on these estimates, the method assigns probabilities of relevance to unjudged documents by solving an optimization problem that minimizes the difference in average precisions obtained by using a true ranking (generated by assessors at TREC) and one generated by inferring the relevance of unjudged documents. It then converts these probabilities into binary judgments of relevance using *randomized rounding*.

The resulting judgments set was shown to bear a high correlation to the actual judgments set.

Jensen presents a technique in [21] for obtaining automatic relevance judgments from web taxonomies. The technique requires a query log that sufficiently represents the query population on the web, and a human-edited taxonomy of documents, that is not biased towards any search-engine. For an informational/topical-search task, all the documents corresponding to a category in the taxonomy whose name exactly matches a query are treated as relevant. For a named-page finding task, this method retrieves all the documents whose human-edited title exactly matches the query. These documents are treated as the 'most-relevant' for the corresponding query. Jensen et al. show that judgments inferred in this way can be used for repeatable and reliable comparisons of IR systems

2.3 Citation Graphs

Networks are interesting structures that can be used for the propagation of various properties from one node to others through link structure. It is useful to be able to identify the 'authority' or the 'relative standing' of nodes who can be used to propagate reliable information through the network. Research in social networks has formulated various ways of determining the 'standing' of a node. Amongst several popular formulations was one defined by Hubbell: The standing s_j of a node j is a solution to the system of equations $s_j = e_j + \sum_i A_{ij} s_i$, where s_j is the standing of a node j, e_j is the prior weight of node j, A_{ij} is the strength of the reference from node i to node j. Much subsequent research on citation graphs can be seen as building on this framework. The field of *bibliometrics* was one of the first to make use of this idea.

Bibliometrics is the study of written documents and their citations. Research in Bibliometrics evaluates the influence of a journal in a network of journal citations. The most popular measure in this field is Garfield's *impact factor*, which counts the number of inlinks to a journal as a measue of its influence. Pinski and Narin [25] made the observation that not all citations may be equally influential; they calculated 'influence weights' based on the idea

that a journal is influential if it is recursively cited by other influential journals. Geller [17] observed that the influence weights correspond to stationary probabilities of a random walk through the network of journal citations. Doreian [15, 16] proposed that the influence weight of a journal essentially corresponds to the standing of a node in a social network (in this case a network of journal citations) and can be obtained by running Hubbell's computations iteratively; the standings computed in one iteration become the prior weights for the next iteration.

Standard citation analysis has been applied to the hypertextual citation structure of the web. Algorithms like PageRank [6] and HITS [23] extended the concept of an academic citation to the notion of a 'link' between 2 web pages. PageRank computes the relative importance of web pages and uses this information to re-rank the seach results for a query. The idea behind PageRank is that an important page is linked to by other important pages, which gives rise to a recursive computation. The PageRank of a page is equal to the sum of the ranks of the pages that point to it weighed by the number of forward links from those pages. The ranking follows the walk of a 'random surfer' in a hyperlinked environment; the surfer starts randomly on some page and successively keeps clicking on links, but eventually gets bored and 'jumps' to a random page with a uniform probability.

PageRank computes 'global importance' scores for web pages, and these scores can be specialized to create a personalized view of importance. Unlike PageRank, that computes a single importance score for every webpage, Haveliwala [19] suggests *customizing* the search results according to a *topic*. The technique initially computes a set of topic-sensitive importance scores for each page offline. During query-processing, the topic-sensitive scores for a page are combined based on the topics encompassed by the query to form a 'composite-PageRank' score for pages matching a query.

Glen and Widom [20] create a set P of user-selected pages that define importance. This set essentially comprises of a subset of the set of hub pages (pages with high PageRank). Instead of randomly jumping to a page with uniform probability as in the PageRank algorithm, the jump is restricted to the set P, increasing the probability that a random surfer will

prefer a page in P's near environment. In essence, this approach creates a personalized view of the pages on the web. The TrustRank [18] algorithm is similar to the above approach, i.e does not assign a uniform prior probability for all web pages. The algorithm initially assigns a non-zero score to some pages that are manually determined to be spam-free. The score of such pages is propagated to pages that are reachable from them to determine their likelihood of being spam-free.

Another popular algorithm that makes use of the hyperlinked structure of the web is HITS. Unlike PageRank, which propagates information from one authority node to another, HITS proposes the flow of information from one authority to another using 'hubs'. An 'authority' node may not endorse another 'authority' node, which is why PageRank uses random jumps to deal with the problem of an authority essentially to nowhere. The algorithm constructs a focussed subgraph of the web with respect to a topic to find 2 sets of pages; 'authority' pages that contain a lot of relavant information on a topic and the 'hub' pages that point to a lot of relevant 'authority' pages.

2.4 Our Approach

The network algorithms outlined above consider a hyperlink as conferring a notion of authority/approval. We will use this idea, but instead of using explicit links, we will use an ad hoc network based on content similarity. Like TrustRank and the personalized versions of PageRank, we will use this network to propagate information about a topic from documents in the network that have been assessed by humans to documents in it that have not been assessed. We will then use this information to infer relevance judgments that we can use to evaluate retrieval systems.

Chapter 3

METHODS

To estimate the relevance of unjudged documents and hence evaluate retrieval systems, we make use of the cluster hypothesis that states: "closely associated documents tend to be relevant to the same requests". If this is true, it suggests that we can leverage some measure of association between unjudged documents and judged relevant documents to identify new relevant documents among those that have not been judged. A natural indicator of association is a hyperlink between two documents; if one document links to another, it suggests those two documents have something in common.

Links alone may not be sufficient indicators of association, however. Many links between web pages exist only because the pages share a domain or for other reasons that have little to do with content or meaning. If we can create new "links" that are based on similarity of content, we can take advantage of existing citation-graph algorithms and the cluster hypothesis to propagate relevance from judged documents to unjudged documents.

We begin this section by describing the PageRank citation-graph algorithms in more detail. We then discuss linking documents by similarity of content and using a refinement of PageRank to propagate human relevance judgments along those links. Section 3.4 presents a complete summary of our approach."

3.1 PageRank

Pagerank[1] is based on a mutual-reinforcement relationship between web pages. It uses the connectivity information between pages to assign popularity scores: a popular page

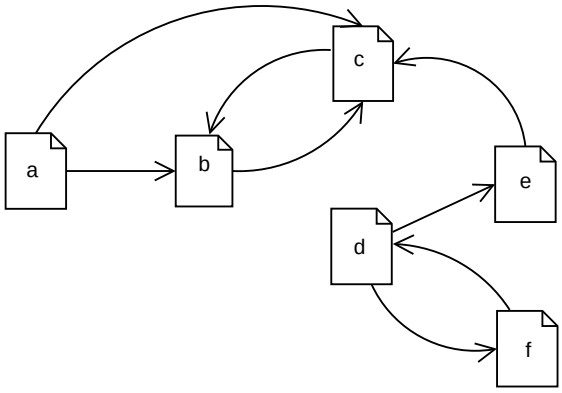


Figure 3.1: An example webgraph

contributes to the scores of other pages, and its own score is a function of the popularity of other pages. This means that PageRank is recursive: calculating the PageRank of a page a requires knowing the PageRanks of the pages that link to a, which could in turn depend on the PageRank of a.

Pages are initially assigned uniform PageRanks of $1/N$, where N is the total number of pages. PageRank is then computed iteratively. The PageRank of a page b on iteration i is the sum of the current PageRanks of each page a that links to b divided by the number of outgoing links from page a. If $G(W, E)$ is a graph in which nodes W are web pages and edges E are hyperlinks between pages, the PageRank of b is computed as:

$$pr(b) = \sum_{(a,b)\in E} \frac{pr(a)}{outlinks(a)} \tag{3.1}$$

where *outlinks(a)* is the number of outgoing links from page a. Figure 1 shows an example web graph. In the figure, a and c link to b. Hence the PageRanks of pages a, b, c are calculated as:

Intial PageRanks:

$$pr(a) = pr(b) = pr(c) = 0.167$$

After the 1st iteration:

$$pr(a) = 0$$

$$pr(b) = pr(a)/2 + pr(c)/1 = 0.167/2 + 0.167,$$

$$pr(c) = pr(a)/2 + pr(b)/1 + pr(e)/1 = 0.167/2 + 0.167 + 0.167$$

Page a links to pages b and c, but no other page links to a. Hence, page a will have a low PageRank. Pages b, c form a loop (Pages d, f also link to each other, but they do not form a loop, as page d links to page e). The PageRanks of pages b and c will end up being higher than those of d and f, since the pair b, c have more inlinks than the pair d, f.

PageRank can be seen as modeling a 'random surfer' visiting web pages in a hyper-linked environment. The random surfer starts at an arbitary page and keeps on following successive links randomly. There is a uniform probability of going to any linked page. In Figure 1, if the random surfer starts at d and goes to f, the surfer might get stuck by clicking on the links from d and f back and forth. But the surfer can break out of the loop by randomly going to e. If the surfer starts at a, and then follows the link to either b or c, there will be no possibility of breaking out of the loop b, c. Thus b and c will have all the probability. This phenomenon is called as "rank-sink".

The computed PageRanks after the 1st iteration are as follows:

$$[a : 0.0, b : 0.25, c : 0.416, d : 0.167, e : 0.083, f : 0.083]$$

after 20 iterations, ..

$$[a : 0.0, b : 0.583, c : 0.416, d : 0.00016, e : 0.00016, f : 0.00016]$$

As we can see, after 20 iterations, pages b and c have nearly all of the probability mass, because the random surfer is always going to end up clicking between those two pages. This loop is called a "rank sink".

We can solve this problem by allowing the random surfer to randomly "jump" to some other page in the web graph without regard to the link structure. These jumps happen with uniform probability (all pages are equally likely landing points), so the probability is $(1-\alpha)/N$. This acts as a "source of rank". The parameter α is determined by the researcher; $\alpha = 0.85$ is a common value.

The PageRank of a page b in a network of N pages is thus defined as:

$$pr(b) = \alpha \sum_{(a,b)\in E} \frac{pr(a)}{outlinks(a)} + \frac{1-\alpha}{N} \tag{3.2}$$

For the example webgraph, we computed the following PageRanks for 20 iterations and α=0.85.

$$[a : 0.025, b : 0.386, c : 0.405, d : 0.072, e : 0.056, f : 0.056]$$

The PageRanks seemed to be changing marginally after the 20th iteration. The correctness of the PageRanks can be confirmed from the webgraph in Figure 1. As page a is not linked to by any other page, it receives the lowest PageRank. Pages e and f have a common inlink from page d, therefore their PageRanks are equal, but low as d has a low PageRank. Pages b and c receive the highest PageRanks, as they endorse each other repeatedly in PageRank iterations and the loop b, c receives rank source from 4 pages, namely a, b, c and e. Also, c receives a slightly higher PageRank than b as it also gets a rank contribution from e, besides a. The PageRanks of all pages in a network (also from the example) sum to 1, as they are essentially probabilities.

3.1.1 Weighted PageRank

A weighted graph is one in which the edges have real-valued weights instead of 0/1 values. A weighted webgraph may be desirable in situations where one link is preferred over another, instead of endorsing all links equally or entirely ruling them out, as in the case of a 0/1 webgraph employed by the tradition PageRank approach. The weights on the links between pages signifies the *strength* of the endorsement from one page to another. One scenario where a weighted webgraph may be useful is for expressing graded relevance; for expressiong the notions of *highly relevant*, *relevant* and *non-relevant*. Also, the component $\frac{\sum_{(a,b)\in E} pr(a)}{outlinks(a)}$ in the PageRank formula propagates an equal portion of the PageRank of b to pages that it links to. This suggests that b equally endorses all outgoing links, which is not desired.

In a *weighted* PageRank [26] technique that operates on a weighted graph, there is a higher probability of endorsing an outgoing link with the largest weight. The calculation would proceed as follows:

$$pr(b) = \alpha \sum_{(a,b)\in E} \frac{w(a,b)pr(a)}{\sum_{(a,c)\in E} w(a,c)} + \frac{1-\alpha}{N} \tag{3.3}$$

3.1.2 Computing PageRank using MapReduce

PageRank can be computationally intensive, so it would be good to parallelize it. MapReduce is a framework for massive parallelization by dividing a computation into many small independent jobs, then recombining the results of the jobs into one final result. For PageRank, we can use it to calculate "partial PageRanks" based on knowing that one page links to another, then obtain final PageRank scores by summing up those partial values. PageRank iterations can be run on the MapReduce framework as follows:

Algorithm 1 PageRank using MapReduce

`procedure map(`p_i`,` $outlinks(p_i)$`):`
`for` $j \leftarrow 1$ `to len(outlinks(`p_i`)) do`
  $pr(p_j) = \frac{pr(p_i)}{outlinks(p_i)}$
  `print` p_j`,` $pr(p_j)$
`end for`

`procedure reduce(`$p_j, (pr(p_{j1}), pr(p_{j2}), ...)$`):`
$pr(p_j) = \alpha * \sum pr(p_{jr}) + \frac{1-\alpha}{N}$
`print` p_j`,` $pr(p_j)$

The input to the `map` procedure is a page-identifier p_i and a vector *outlinks(p_i)* containing the outgoing links from p_i. This phase outputs key-value pairs of type *(page-identifier, partial -PageRank)* for each p_j that p_i links to.
The `reduce` operator collects all tuples with a key p_j and computes a final PageRank from the partial PageRanks. Ideally, the iterations must be run to convergence, but only a fixed number of iterations are run in practice.

This will ensure that the human judgments are propagated according to the strength of association.

3.2 Similarity

The edges in a weighted webgraph can be modelled using any metric. Content similarity between documents can be used as one such metric. We model content using a "bag-of-words" description of a document, and a document is modeled as a vector in $\mid V \mid$-dimensional space, where V is the vocabulary. Each term in a document has a weight associated with it. Then we can calculate a measure of content similarity as the cosine of the angle between two vectors.

$$sim(a,b) = cos(a,b) = \frac{\sum_{t \in V} w_{(a,t)} w_{(b,t)}}{\sqrt{\sum_{t \in V} w_{(a,t)}^2} \sqrt{\sum_{t \in V} w_{(b,t)}^2}} \tag{3.4}$$

We used $w_{(a,t)} = \log(tf_{(a,t)} + 1) \log N/df_t$, where $tf_{(a,t)}$ is the number of times term t appears in page a and df_t is the total number of pages t appears in. This gives greater weight to terms that appear more frequently in a document, but less weight to terms that appear in many documents. Term weights are usually a function of term frequency in a document and document frequency in a collection.

3.3 TrustRank

A second drawback with the PageRank algorithm is that it assigns an equal (prior) score to all pages. This means that all pages are initially considered to be of equal importance.

TrustRank, a network algorithm that separates good pages from spam, incorporates a *biased* PageRank technique, that assigns a non-zero score to a set of manually identified 'good'(spam-free) pages.The score of these 'good' pages is then propagated in subsequent iterations to pages they point to, to find other pages that may be good.

The TrustRank algorithm is outlined below.

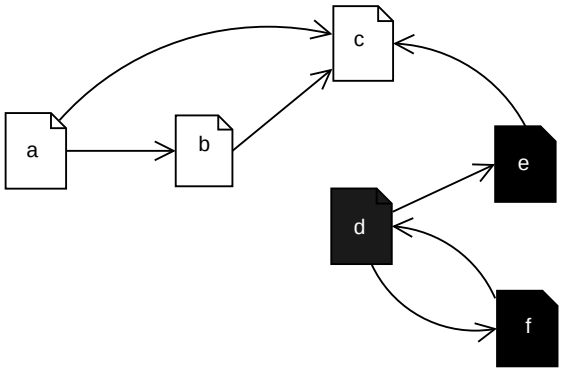


Figure 3.2: A hypothetical web-model of good and spam (black) pages

Algorithm 2 TrustRank algorithm

1: Construct a $\mathbf{seed-set}$ of documents.
2: $\mathbf{d} = 0_N$
3: **for each** $s \in$ seed-set **do**
4: $\quad \mathbf{d}[\mathrm{s}] = human\ judgment : 1(good)/\ 0(bad)$
5: **end for**
6: $d = d/\sum d$
Compute TrustRank scores as follows:
7: $\mathbf{tr} = \mathbf{d}$
8: **for** $i \leftarrow 1$ to M **do**
9: $\quad$ **for each** page b **do**
10: $\quad\quad tr(b) = \alpha \sum_{(a,b)\in E} \frac{tr(a)}{outlinks(a)} + (1-\alpha)\ d_b$
11: $\quad$ **end for**
12: **end for**

- The algorithm begins by selecting a set of documents, called the *seed-set* that are manually examined by an expert to be reputable or bad. The size of this set is kept small to limit the amount of human effort required in this step.

- Step(2) initializes a one-dimensional static score distribution vector $\mathbf{d}$ of size N; entries in d corresponding to manually judged *good*-seed pages receive a score of 1, *bad*-seed pages receive a score of 0.

- Step(3) normalizes the $\mathbf{d}$ vector, so that its entries sum to 1.

- Finally, the prior scores of *good* pages identified in step 2 is propagated to other pages that are reachable from them in M steps/iterations. In each iteration, the TrustRank score of a page is dampened by a factor of α and equally split among the pages it points to. The authors of the paper used $\alpha = 0.85$ and found M = 20 iterations to be sufficient in achieving convergence in the relative ordering of the pages.

The seed-set of pages is essentially identifying a set of pages that will be most useful in propagating a desired property in a network of pages. TrustRank propagates spam judgments to identify and separate spam pages. The seed-set could also consist of relevance judgments, topicality judgments used by [19], preference judgments used by [20] or any other kind of human judgment of a property of a page.

3.3.1 Weighted TrustRank

The edges(links) in the TrustRank web-model could be assigned weights, just as in the weighted PageRank technique outlined in section 3.1.1, to reflect the desirability of one link over the other. This will ensure that the human judgments are propagated according to the strength of association. Accordingly, *weighted* TrustRank scores can be computed in step(9) of the TrustRank algorithm as:

$$tr(b) = \alpha \sum_{b} \frac{w(a,b)\ tr(a)}{\sum_a w(a,c)} + (1-\alpha)\,.\,d \tag{3.5}$$

3.4 Our Approach

Our goal is to infer the relevance of unjudged documents in light of incomplete judgments and use it to evaluate systems. TrustRank propagates spam judgments through a network of web pages to identify other spam-free pages based on their *connectivity* with the *good* seed pages. Similarly, we use the *weighted* TrustRank technique introduced in section 3.3.1 to propagate relevance information from some manually judged relevant pages to predict the relevance of unjudged documents based on their *degree of similarity* with the judged

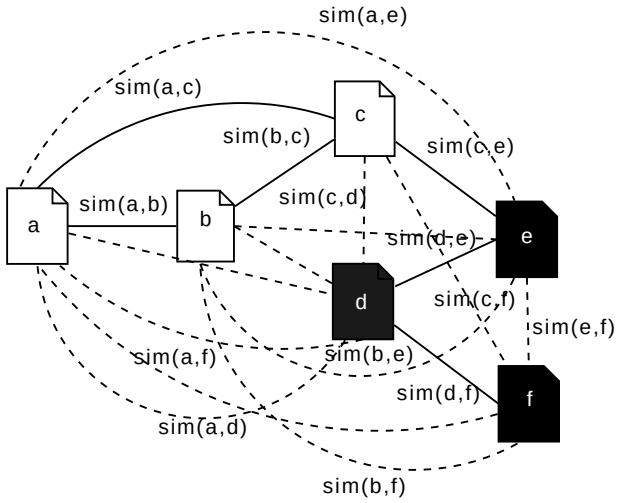


Figure 3.3: A web-model of relevant and non-relevant (black) pages in the *weighted* TrustRank approach

relevant documents. We model the document collection as an *undirected weighted-graph*, in which the documents constitute the vertices and the notion of an edge/link between two documents a and b is the degree of cosine similarity, $sim(a, b)$ between them. Taking similarity between every pair of documents into account gives rise to a link between every document pair in the webgraph (shown in Figure 3 using dotted lines). Also, $sim(a, b) = sim(b, a)$, which is why the webgraph is considered undirected. A hypothetical collection webgraph is shown in Figure 3.

3.4.1 Seed-set Construction

We use the following seed-selection techniques to find whether any particular technique influences TrustRanks greater than the other:

(a) Pooling: a method commonly used to build the test collections for TREC tasks, to build our seed-set of documents: the top k documents ranked by different systems are pooled and included in the seed-set. We want to keep k small, so that extensive human effort

is not required in determining the relevance information of these documents in step(2) of the algorithm.

(b) Statistical Sampling [3]: This method also examines a ranked list of documents for inclusion into the seed-set, but samples at those locations "where the relevant documents are most likely to be found", which would be at the top of a ranked list.

The method employs a non-uniform sampling strategy, i.e samples documents from the top of a ranked list with higher probability, and samples documents from the bottom of the list with a lower probabaility. We also ensure that using this strategy, we include the same-number of documents in the seed-set as obtained with *pooling*.

3.4.2 Static Prior Score Distribution

We use the term **prior** for the static score distribution vector **d** from the TrustRank algorithm. The documents in the seed-set are examined by a human expert to be relevant (rel=1) or non-relevant (rel=0), and are accordingly assigned prior scores of $rel = 1$ or $rel = 0$. Documents outside the seed-set are treated as unjudged documents, and are given a prior score of $rel = 0.5$.

The **prior** vector initialized in step(2) is normalized as follows:

$$\mathbf{d} = prior = \frac{rel}{\sum rel_i} \tag{3.6}$$

3.4.3 Mapping TrustRanks to Relevance Judgments

After running algorithm 2, each document has a TrustRank score based on its similarity to relevant documents in the seed set. We then need to convert TrustRank scores to binary relevance judgments. We will do that by ranking the documents in decreasing order of TrustRank score, then finding a threshold t for TrustRank such that all documents with scores above t are considered relevant and all documents with scores below t are considered nonrelevant.

We normalized the TrustRank scores as they were not in the same scale from query to query. For intra-query normalization (required for computing a macro-averaged F), we normalized the TrustRanks querywise and for inter-query normalization (for computing micro-averaged F), we normalized the TrustRanks over queries.

$$normalized\ tr = \frac{tr - tr_{min}}{tr_{max} - tr_{min}} \tag{3.7}$$

where tr stands for a TrustRank score, tr_{min} and tr_{max} are the *query* mininum and maximum TrustRanks in case of intra-query normalization and *global* minimum and maximum in case of inter-query normalizaion.

3.4.3.1 F-measure

We use the 'F-measure' evaluation metric to learn a threshold (hereafter referred to as *TrustRank threshold*) to separate relevant documents from non-relevant ones. F-measure is the weighted harmonic mean of precision and recall and is defined as:

$$F - measure = \frac{2\,.\,precision\,.\,recall}{precision + recall} \tag{3.8}$$

Precision and *recall* are defined as :

$$precision = \frac{\#\text{ of relevant docs with score} > t}{\#\text{ of docs with score} > t} \tag{3.9}$$

$$recall = \frac{\#\text{ of relevant docs with score} > t}{\#\text{ of relevant docs}} \tag{3.10}$$

where t is the TrustRank threshold. Since we are training a threshold on the seed-set of documents, the number of relevant documents is the number in the seed set only. We hence learn a threshold that will give us best prediction results for the documents in the seed set.

We compute the F-measure over the TrustRanks using the following 2 kinds of averages:

1. Micro-averaging: In this technique, F-measure is averaged over the TrustRank scores of all queries Q. The algorithm for computing micro-averaged F-measure and the corresponding micro-averaged threshold (t-micro) is shown below. A document is classified as relevant if its weighted TrustRank is greater than or equal to *t-micro*.

Algorithm 3 Computing micro-averaged F-measure

Require: vector **tr** of TrustRanks, normalized over all queries
  t = 0
  while $t \leq 1$ **do**
    A = $\sum_{q \in Q}$ # of docs i s.t. i is relevant to q and $tr_{i,q} < t$
    B = $\sum_{q \in Q}$ # of docs i s.t. i is relevant to q and $tr_{i,q} \geq t$
    C = $\sum_{q \in Q}$ # of docs i s.t. i is non-relevant to q and $tr_{i,q} < t$
    D = $\sum_{q \in Q}$ # of docs i s.t. i is non-relevant to q and $tr_{i,q} \geq t$
    $precision_t = \frac{B}{B+D}$
    $recall_t = \frac{B}{B+A}$
    $F_t = \frac{2 . precision_t . recall_t}{precision_t + recall_t}$
    t = t + 0.05
  end while
  t-micro $\leftarrow$ arg $\max_t F_t$

2. Macro-averaging: Here, F-measure is calculated over each query $q \in Q$ first and then averaged over all queries.

   A document is classified as relevant if its weighted TrustRank is greater than or equal to *t-macro*.

As Ozgur et al. [24] point out, the micro-averaged F-measure is influenced by the classifier's performance on common classes, and the maco-averaged F-measure is influenced by the classifier's performance on rare classes.

3.4.3.2 Logistic Regression

Logistic Regression consists of identifying a relationship between two variables X and Y; where X = $< X_1 ... X_n >$ is a vector of continuous/discrete variables and Y is a discrete

Algorithm 4 Computing macro-averaged F-measure

Require: vector **tr** of TrustRanks, normalized querywise
  t = 0
  while $t \leq 1$ **do**
    for $q \leftarrow 1$ to $\#Q$ **do**
      A = # of docs i s.t. i is relevant to q and $tr_{i,q} < t$
      B = # of docs i s.t. i is relevant to q and $tr_{i,q} \geq t$
      C = # of docs i s.t. i is non-relevant to q and $tr_{i,q} < t$
      D = # of docs i s.t. i is non-relevant to q and $tr_{i,q} \geq t$
      $precision_{t,q} = \frac{B}{B+D}$
      $recall_{t,q} = \frac{A}{B+A}$
      $F_{t,q} = \frac{2\,.\,precision_{t,q}\,.\,recall_{t,q}}{precision_{t,q}+recall_{t,q}}$
    end for
    $Favg_t = \frac{\sum_{q \in Q} F_{t,q}}{|Q|}$
    t = t + 0.05
  end while
  t-macro $\leftarrow \arg\max_t Favg_t$

response variable. The vector X represents a set of independent variables that can influence the outcome in a strong/weak way, and Y represents the probability of a particular outcome. Our approach models X_i as the TrustRank for document i and Y as its relevance judgment. The probability distribution $P(Y|X)$ follows the distribution shown in Figure 4. As Y is boolean(relevant/non-relevant), $P(Y|X)$ is defined as:

$$P(Y|X) = \frac{1}{1 + e^{(\beta_0 + \sum_i^n \beta_i X_i)}} \tag{3.11}$$

and:

$$P(Y = 1|X) = \frac{1}{1 + e^{(\beta_0 + \sum_i^n \beta_i X_i)}} \tag{3.12}$$

$$P(Y = 0|X) = \frac{e^{(\beta_0 + \sum_i^n \beta_i X_i)}}{1 + e^{(\beta_0 + \sum_i^n \beta_i X_i)}} \tag{3.13}$$

Taking log on both sides of equation (1) results in a simple linear classification rule that assigns a judgment of Y = 0/non-relevant if:

$$0 < \beta_0 + \beta_i \sum_i^n X_i \tag{3.14}$$

where β_0 is the intercept and $\sum_i^n \beta_i$ are the regression coefficients, Y = 1 otherwise.

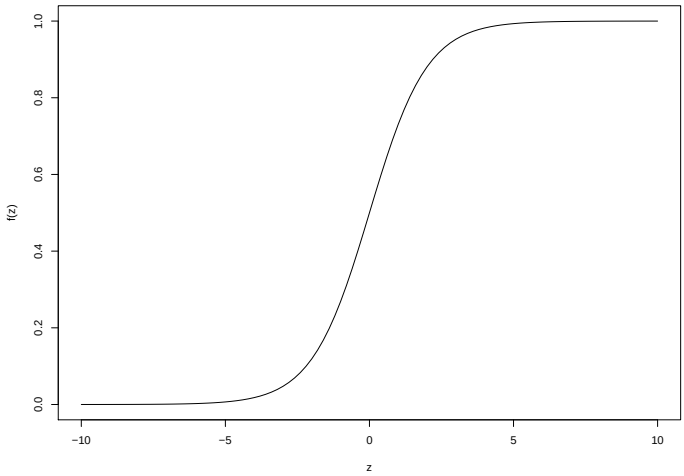


Figure 3.4: Logistic function assumed by $P(Y|X)$

We will create a new set of relevance judgments consisting of the seed set and the predicted relevant documents from either the threshold classifier or the logistic regression classifier.

Chapter 4

EXPERIMENTS

The following section describes our test collection, the experimental procedure and the implementation details. First, we construct an adhoc network of document similarities of the documents in the collection. We then select a seed-set of judged documents that would be the most desirable in propagating relevance through the document-similarity network. Thereafter, we train weighted TrustRank thresholds (outlined below) on both the judged and unjudged documents and infer the relevance of unjudged documents. We use the inferred judgments together with the seed-set of judgments to evaluate systems.

4.1 Data

We will need a test collection (consisting of documents, queries, and relevance judgments) that we can use to simulate our method, as well as some actual retrieval runs that we can evaluate using our method. We evaluate the performance of our approach over the runs submitted to the TREC Terabyte tracks [11, 13, 8] and the 2008 Million Query (MQ) track [2]. Hence our test collection consists of the *document* collection, the set of *topics* and the *relevance judgments* used by the above tracks.

The Terabyte track was a competition to evaluate retrieval performance on a large corpus of 25 million web pages (the largest available at the time the Terabyte track started). The Million Query track was also for evaluating retrieval performance over a large collection, but another goal was to evaluate evaluation techniques.

4.1.1 Corpora

We used the GOV2 corpus as it was the document collection used with the above tracks. It contains a crawl of the pages in the '.gov' domain and is a mix of plain text, HTML and the extracted text of pdf, word and postscript files. It is 426 GB in size and contains a total of 25 million documents.

4.1.2 Queries

The TREC Terabyte tracks used TREC topics 701-850, which consist of:

1. *title* field, typically a keyword query
2. *description* field that provides a slightly longer statement of the topic requirements
3. *narrative* that describes all other information required in a short paragraph

The different groups had the choice to use any or all of the topic fields when creating queries from the topic statements. However, each group was required to a submit an automatic run that used only the title fields.

The 2008 MQ queries were 10000 title-only topics which were sampled from an Internet search engine, of which 782 were converted to topic statements and judged.

4.1.3 Runs

The Terabyte and the MQ tracks attarcted participation from both universities and industries alike. Notable ones included University of Massachusettes at Amherst, Carnegie Mellon University, IBM and Yahoo!. For the Terabyte tracks, the groups were permitted to submit up to 5 runs. Each run consisted of the top 10000 documents retrieved for every topic. Each run in the 2008 MQ track retrieved the top 1000 documents. Table 1 provides a summary of the runs in the Terabyte and 2008 MQ tracks.

TREC topics	avg.#judgments query	avg.#rel/ query	#participating groups	#submitted runs
701-750	1185.244	216.673	17	70
751-800	905.82	208.14	18	58
801-850	639.68	117.86	20	80
2008 MQ topics	19.451	0.391	7	25

Table 4.1: Summary of runs

4.2 Experimental Setup

We performed 2 sets of experiments. The first experiment consists of training weighted TrustRank thresholds on seed-set of documents from the Terabyte runs and using the thresholds to make relevance predictions for unjudged documents (documents outside the seed-set) from the same runs. The thresholds are trained and tested on the same set of queries, but the training and the test sets consist of specific samples from the collection (of judged and unjudged documents respectively). Hence, this experiment illustrates *transductive learning*.

The second consists of training weighted TrustRank threholds on seed-set of documents from Terabyte runs and making relevance predictions for documents from the 2008 MQ runs. As we use the thresholds trained from one set of queries on another, this experiment illustrates *semi-supervised learning*.

For each type of experiment the following setting was common:

1. Construction of a similarity network of all documents in the collection
2. Building a seed-set of (judged for relevance) documents that will be used to propagate relevance through the similarity network
3. Training weighted TrustRank thresholds on the seed-set to infer relevance judgments of unjudged documents

4.3 Experimental Procedure

1. *Seed-set construction* :

(a) With pooling, we retrieved the top n documents from every run submitted to the TREC Terabyte tracks; and included only those documents in the seed-set which had a relevance judgment in the original TREC ranking (hereafter called *original-qrels*) for that track. So, the documents included in the seed-set are essentially a subset of *original-qrels* (and hence referred to as *subset-qrels*) and used as training data. All other documents outside the seed-set and part of *original-qrels* were treated as unjudged, and included in the test data. For our experiments, we tried pooling depths of $k = 5$, 10 and 25.

(b) With *statistical sampling*, we sampled the documents from a pool of depth-50 from all the retrieval runs with a higher probability, and decreased the probability of sampling documents from outside the pool. Accordingly, we constrcuted 3 different seed-sets by using 5, 10 and 25 samples.

2. *Build network of document similarities*: We compute cosine similarities for each pair of documents part of both the training and the test sets. We use the document similarity matrix thus obatained as the link structure for computing TrustRanks.

3. *Training TrustRank on seed-set*: As mentioned in section 3.4.2, we initialized the scores for the judged relevant documents in the seed-set to 1, judged non-relevant documents to 0. Unjudged documents are given a prior score of 0.5. We used $\alpha = 0.85$ in our experiments.

4. *Making relevance predictions for unjudged documents*:

   (a) TrustRank-threshold: We trained 3 kinds of classifiers in this category: We had to normalize TrustRanks for computing the following classifiers as they were very small in magnitude; some of them small to the order of e^{-5}.

      i. *t-macro*: We normalized TrustRanks querywise and apply the computed t-macro to classify documents as relevant/non-relevant.

      ii. *t-micro*: We normalized TrustRanks over queries and applied t-micro on the test set of documents.

iii. *local-t-micro*: In this approach, we normalized TrustRanks querywise and applied t-micro for classification.

(b) Logistic Regression: We fit the TrustRanks and relevance labels of training documents from Terabyte queries to the logistic functon and predicted probabilties of relevance for documents from the Million Query runs

5. *Extending original-qrels*: We extend our seed-set of documents (subset-qrels) with the inferred judgments for unjudged documents *inferred-qrels* and call it *extended-qrels*. We will use this extended set to evaluate the runs submitted to the TREC Terabyte and the 2008 Million Query tracks.

4.4 Implementation Details

For building the adhoc network of document similarities, the cosine similarities between all the documents were precomputed and stored on disk, with the documents retrieved from an Indri index. The computation was parallelized query-wise using Grid Engine running on an 8-node-dual-processor-quad-core compute cluster. We used our own map-reducisized implementation of weighted Trustrank written in Python, that executes on the Hadoop streaming [1] framework. Hadoop streaming is a utility that allows any kind of excecutable script as mapper and/or reducer.

We used R's glm function for computing classification using Logistic Regression. We used trec_eval, a standard evaluaton tool at TREC developed by Chris Buckley, to evaluate a set of runs against relevance judgments. For computing EMAP of a run, we used the `mtc-eval` evaluation tool developed by Carterette et al.

4.5 Evaluation

We want to measure our classifier's ability to classify an unjudged document as relevant/non-relevant and use the classified data to evaluate retrieval systems.

4.5.1 Evaluating Classification Performance

1. **Classification using *TrustRank-threshold***: We used *t-macro*, *t-micro* and *local-t-micro* to classify unjudged documents for relevance. These relevance predictions form the inferred-qrels which is appended to subset-qrels for evaluating retrieval runs in the TREC Terabyte and 2008 Million Query tracks. We then evaluate the performance of the classification task by computing precision and recall

$$precision = \frac{num\_actually\_rel \ \& \ predicted\_rel}{num\_predicted\_rel}$$

$$recall = \frac{num\_actually\_rel \ \& \ predicted\_relevant}{num\_actually\_rel}$$

   Precision is the proportion of documents predicted to be relevant by inferred-qrels that are originally relevant. Recall is the proportion of relevant documents retrieved using inferred-qrels of the total relevant documents

2. **Classification using Logistic Regression**: The TrustRanks and relevance labels of documents in the training set are fit to a logistic curve to make predictions of relevance for test data. We infer the relevance judgments of the test documents using the following logistic regression classifier (henceforth called *logreg*) rule:

$$i \ is \ non-relevant \ if \ \frac{(1-p_i)}{p_i} > 1, \ else \ i \ is \ relevant \tag{4.1}$$

   where p_i is the prediction for document i.

   We append these inferred judgments to the seed-set of judgments to obtain extended-qrels. We evaluate the classification performance of the extended-qrels using precision and recall.

4.5.2 Evaluating Ability to Evaluate Systems

We compare the rankings generated by original-qrels , subset-qrels and extended-qrels using Kendall's τ correlation coefficient. original-qrels is the set of relevance judgments

obtained from TREC, subset-qrels consists of the seed-set of documents we select for training purposes, and extended-qrels consists of judgements from subset-qrels and inferred relevance judgments for unjudged documents. Kendall's τ counts the number of pairwise disagreements between two lists. The value for τ ranges from [-1,1]; 1 indicates that the two lists are identical and hence in perfect aggreement, 0 indicates that the two lists agree on half of the pairs, and -1 indicates perfect disaggreement.

Chapter 5

RESULTS

5.1 Evaluating Classification

Using the pooling method, we constructed seed-sets pooled to depths of 5,10 and 25. The classification performance of the TrustRank thresholds and the logreg rule on the corresponding seed-sets are presented in the tables below.

As the tables show, precision decreases from left to right in a row, while recall increases from left to right. This means that t-micro and local-t-micro predict many more documents to be relevant than t-macro. Increase in pool-depth also seems to influence the classification results; precision and recall decrease with the pool-depth. This suggests that a deeper pool negatively affects TrustRank, possibly because the proportion of nonrelevant documents increases and overcomes the information contained in the relevant documents.

The F-measure values and hence the classification results computed by training on seed-sets constructed using pooling were slightly better than to those generated using statistical sampling. This suggests that the method for choosing the seed-set seems to make a difference in learning the classifier.

Figures 5.1 and 5.2 show the macro-averaging and micro-averaging process for pool depths of 5, 10 and 25 respectively. In general, as t increases, precision increases, recall decreases. Greater precision leads to decreased recall and greater recall leads to decrease in precision, and this is reflected in the value of F-measure that takes both precision and recall into account. For both sets of plots, precision starts off with a higher value for a shallower pool-depth, recall is high for some time and starts dropping down, but the variation in both

	t-macro		*t-micro*		*local-t-micro*		*logreg*	
run	precision	recall	precision	recall	precision	recall	precision	recall
Terabyte 2004	0.304	0.365	0.274	0.457	0.130	0.482	0.308	0.022
Terabyte 2005	0.329	0.305	0.279	0.620	0.187	0.557	0.315	0.187
Terabyte 2006	0.279	0.321	0.185	0.730	0.134	0.446	0.232	0.483

(a) Classification performance on a depth-5 pooled seed-set

	t-macro		*t-micro*		*local-t-micro*		*logreg*	
run	precision	recall	precision	recall	precision	recall	precision	recall
Terabyte 2004	0.278	0.295	0.260	0.329	0.124	0.416	0.234	0.008
Terabyte 2005	0.304	0.262	0.269	0.504	0.177	0.487	0.292	0.127
Terabyte 2006	0.248	0.238	0.165	0.522	0.116	0.322	0.211	0.309

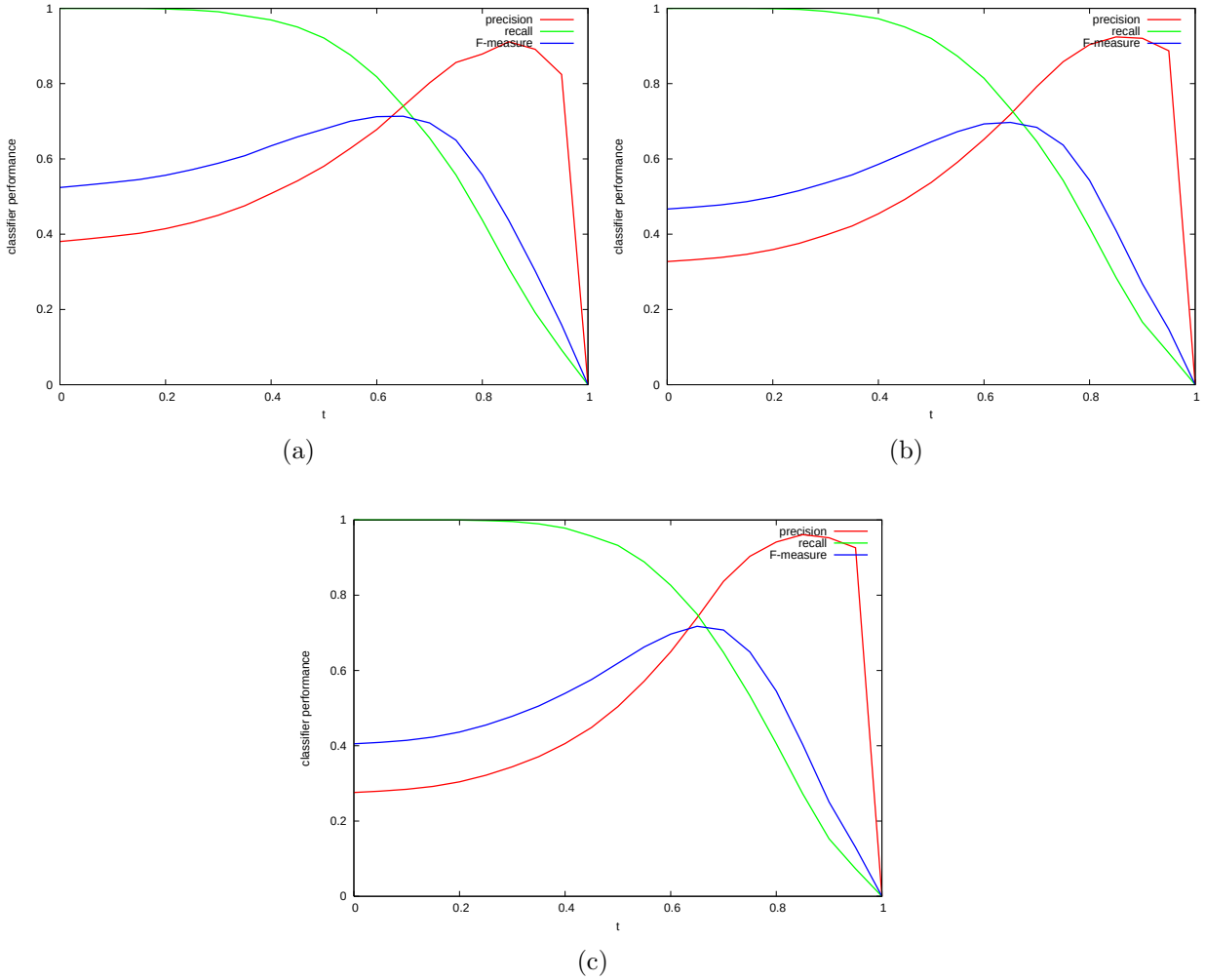
(b) Classification performance on a depth-10 pooled seed-set

	t-macro		*t-micro*		*local-t-micro*		*logreg*	
run	precision	recall	precision	recall	precision	recall	precision	recall
Terabyte 2004	0.250	0.186	0.222	0.209	0.115	0.301	0.098	0.001
Terabyte 2005	0.284	0.191	0.247	0.368	0.160	0.370	0.274	0.072
Terabyte 2006	0.211	0.115	0.135	0.274	0.101	0.185	0.160	0.131

(c) Classification performance on a depth-25 pooled seed-set

Table 5.1: Classification performance produced by training on seed-sets pooled to depths of 5, 10 and 25. Precision and recall increase with the pooled depth. Among the machine-learnt thresholds, *local-t-micro* seems to perform poorly compared to *t-macro* and *t-micro*. logreg does not do well either.

.



(a) (b)

(c)

Figure 5.1: Macro-averaging process on a (a) depth-5, (b) depth-10 and (c) depth-25 pool

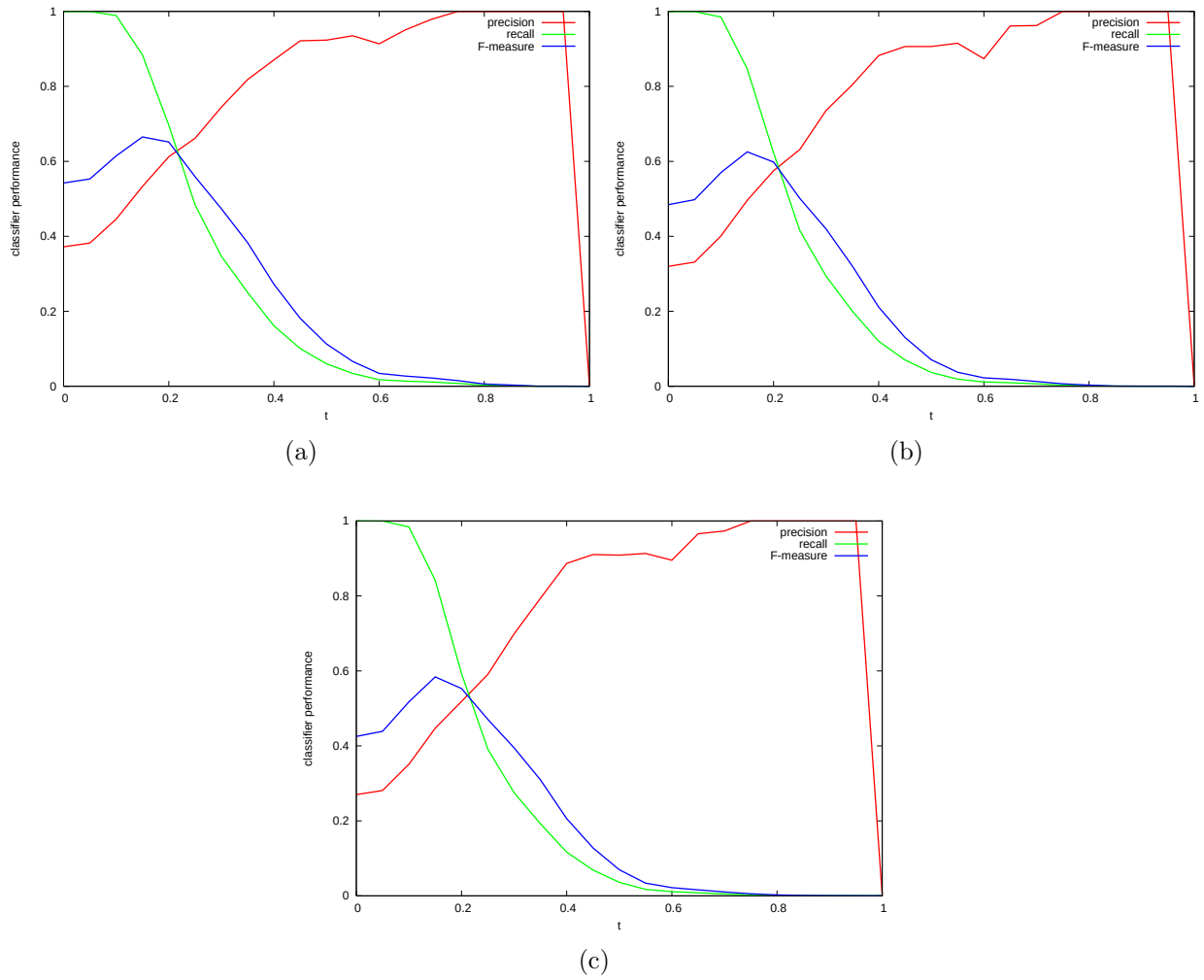


Figure 5.2: Micro-averaging process on a (a) depth-5, (b) depth-10 and (c) depth-25 pool

precision and recall remains more or less the same, which is why the macro/micro averaged F was the same for all the pool depths.

Also, recall starts dropping faster during micro-averaging than macro-averaging. This justifies the statement in section 3.4.3 that micro-averaged F is heavily influenced by the classifier's performance on common categories, and macro-averaged F is influenced by the classifier's performance on rare categories. For any topic, non-relevant documents are more common than relevant documents and as non-relevant documents get a lower TrustRank score than relevant documents, micro-averaged F is lower than the macro-averaged F.

5.2 Evaluating the Evaluation

For both sets of experiments, we extend the training set(subset-qrels) with the relevance predictions generated by the classifiers described above. We use Kendall's τ to measure the correlation between the true ranking (original-qrels) and predicted ranking (extended-qrels).

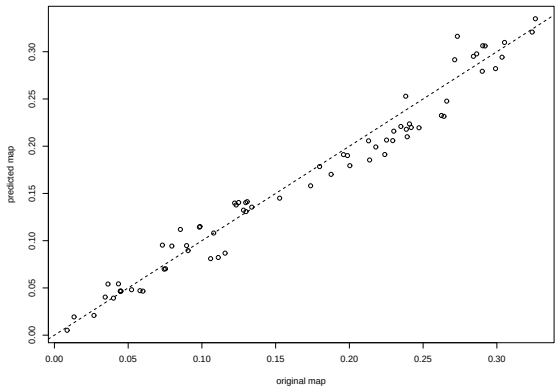
5.2.1 Terabyte Results

Using `trec_eval`, we evaluated the Terabyte-track runs against extended-qrels to compute the corresponding MAP scores of the runs (referred to as *predicted_map* in the plots). In the Figure 7, we plot the correlation between the original maps (`trec_eval` using original-qrels) and the predicted maps in the figures below for the seed-sets obtained using pooling.

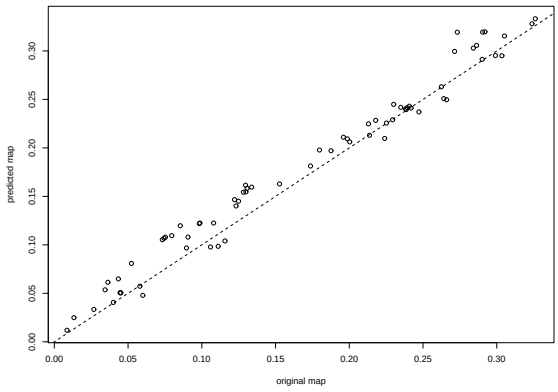
The TrustRank thresholds trained on seed-sets constructed using statistical sampling were identical to those obtained using pooling, with a minor difference in the F-measure values. Hence, the evaluation results using statistical sampling were more or less similar to results obtained using pooling.

It is evident from the plots that there is not much difference in evaluation performance by using *t-macro* or *t-micro* (even though they give very different results for classification). This may be due to the thresholds wrongly predicting the relevance of many documents and still doing well when the judgments are actually used for evaluation. Also, it is not very surprising that τ increases as the pooling-depth/number of samples increases.

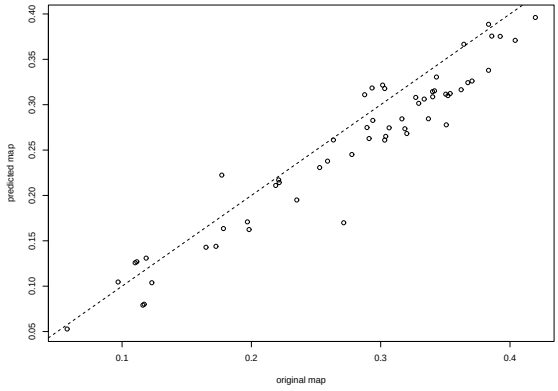
Figure 8 shows a comparison of the performance of subset-qrels, t-macro (pooling), t-micro (pooling) and t-macro (statistical sampling). As the plots suggests, t-macro and t-micro outperform subset-qrels on the Terabyte 2004 and 2006 runs. Also, t-macro-stat-sampling does better than t-macro-pooling on the Terabyte 2005 runs. None of the thresholds did well on the Terabyte 2005 runs.



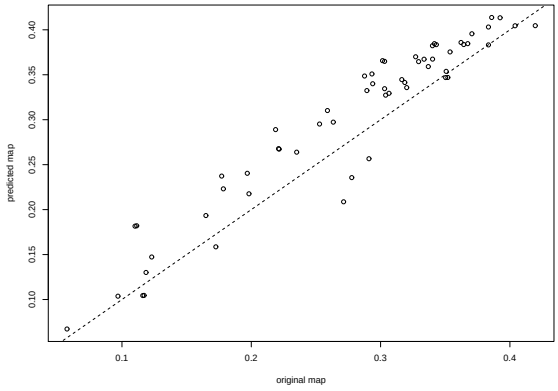
(a) i. Kendall's $\tau = 0.902$



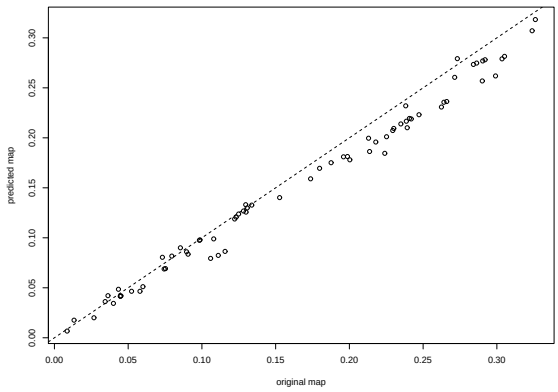
(b) ii. Kendall's $\tau = 0.903$



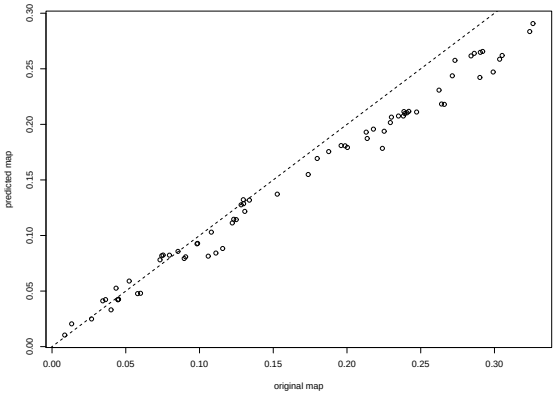
(c) i. Kendall's $\tau = 0.803$



(d) ii. Kendall's $\tau = 0.813$



(e) i. Kendall's $\tau = 0.894$



(f) ii. Kendall's $\tau = 0.877$

Figure 5.3: Correlation between the true ranking and predicted ranking (a & b) depth-5 pooled seed-set of runs from the Terabyte 2004 track, (c & d) depth-10 pooled seed-set of runs from the Terabyte 2005 track, (e & f) depth-25 pooled seed-set of runs from the Terabyte 2006 track. (i & ii) correspond to using *t-macro* and *t-micro* as classifiers.

5.2.1.1 Analysis

We performed a multi-factor ANOVA analysis on the Terabyte results. Considering the τ correlation as a response variable and 'pool-depth'/'number of samples', 'thresholding method' and 'seed-selection technique' as the independent variables, we conclude that:

1. pool-depth/number of samples has a strongly significant effect on τ
2. thresholding (t-macro/t-micro vs local-t-micro) method has a strongly significant effect on τ
3. seed selection algorithm does have a significant effect on τ

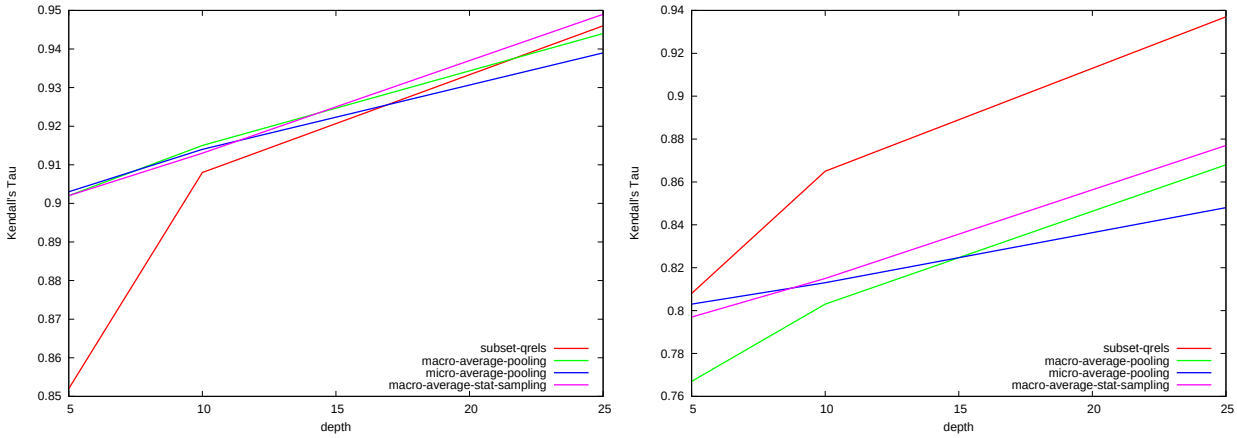
Thus we recommend using a seed set from a pool with depth selected according to budget constraints, and using a macro-average threshold classifier to predict relevance.

5.2.2 MQ Results

We computed weighted TrustRank for documents part of subset-qrels pooled to depth 10, and fit the TrustRanks and relevance labels of the training data to a logistic function to generate probabilities of relevance for the MQ documents.

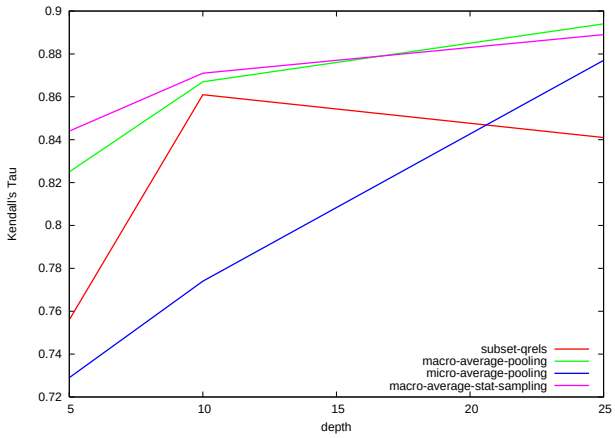
Figure 9 shows a histogram of the relevance predictions for MQ data. It shows a lot of documents being increasingly predicted relevant, which suggests that TrustRank results obtained from one set of queries may not be useful for another set of queries. This is also justified by the low τ score of 0.123 between the true and predicted rankings. We plot the correlation between the 2 rankings in Figure 10.

We used EMAP [10] to evaluate the performance of the predicted ranking for MQ runs and compare these to the EMAPs of the baseline run using Kendall's τ.



(a)

(b)



(c)

Figure 5.4: Change in Kendall's τ in response to increase in pooling depth for *subset-qrels*, *t-macro (pooling)*, *t-micro (pooling)* and *t-macro (statistical sampling)* on the (a) 2004, (b)2005 and (c)2006 Terabyte runs

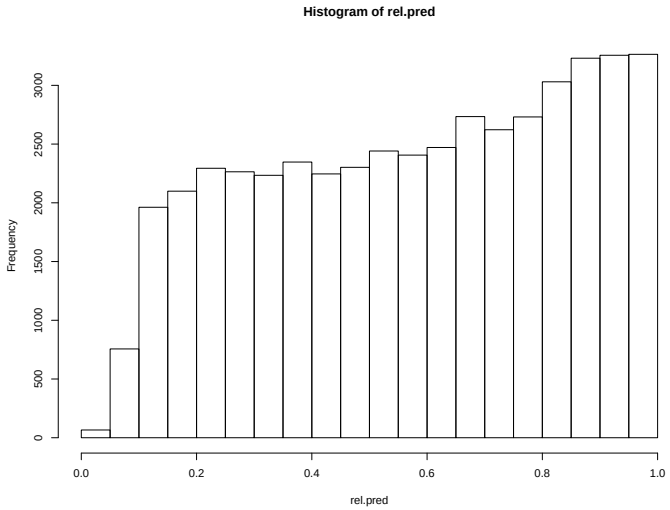


Figure 5.5: Histogram of predictions generated using Logistic Regression for test data from the 2008 MQ track. Y-axis indicates the frequency of retrieved documents and X-axis indicates the probability of relevance.

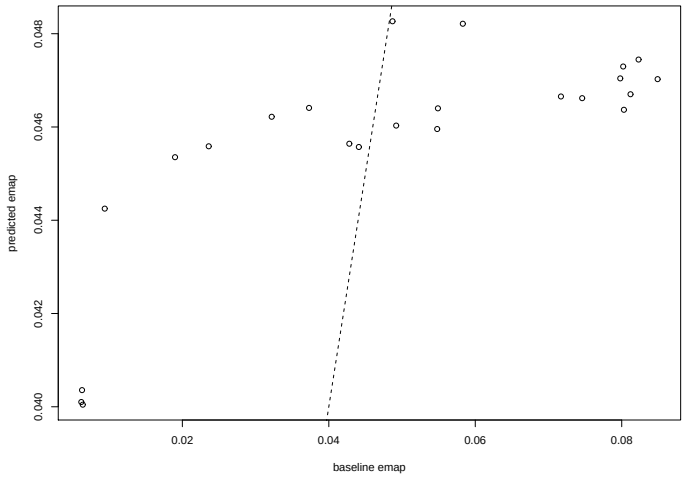


Figure 5.6: Correlation of the predicted EMAPs with the baseline EMAPs. It can be seen that there is almost little to no correlation between the 2 rankings.

Chapter 6

CONCLUSION AND FUTURE WORK

We address the problem of large-scale retrieval evaluation using incomplete relevance judgments. Our proposed solution is a semi-automatic technique that starts by obtaining relevance judgments for a few documents and propagates the relevance information from the judged relevant documents through a document-similarity network to infer the relevance of other unjudged documents in the collection. We then create a complete set of judgments by combining the original seed-set of judgments with the inferred judgments and use it to evaluate the relative performance of different retrieval systems. We believe that our work is a first attempt in solving an evaluation problem by propagating relevance information through document networks.

It is evident from our results that we can effectively compare the performance different retrieval systems by using only few judgments and still achieve a high correlation with the true ranking of retrieval systems. However, our approach is limited in that it cannot learn a relationship between TrustRank scores and relevance for unseen queries. We can only use it to find new relevant documents for queries that we already have judgments for. Nevertheless, we believe this will be a useful technique for rapid development of new test collections.

As part of future work, different similarity measures/term weighting approaches could be used as weights in a weighted webgraph. It would be particularly interesting to explore similarity measures that capitalize on the link structure as well as the document content. It would also be interesting to adapt the 'HITS' approach to similarity graphs.

BIBLIOGRAPHY

[1] Hadoop streaming. http://hadoop.apache.org/common/docs/r0.15.2/streaming.html.

[2] J. Allan, J. A. Aslam, B. Carterette, V. Pavlu, and E. Kanoulas. Million query track 2008 overview. In *Proceedings of TREC 2008*, 2008.

[3] J. A. Aslam and V. Pavlu. A practical sampling strategy for efficient retrieval evaluation. May 2007. Working draft available at the following URL: http://www.ccs.neu.edu/home/jaa/papers/drafts/statAP.html.

[4] J. A. Aslam, V. Pavlu, and E. Yilmaz. A statistical method for system evaluation using incomplete judgments. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 541–548, New York, NY, USA, 2006. ACM.

[5] J. A. Aslam and E. Yilmaz. Inferring document relevance from incomplete information. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 633–642, New York, NY, USA, 2007. ACM.

[6] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, 1998.

[7] C. Buckley and E. M. Voorhees. Retrieval evaluation with incomplete information. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 25–32, New York, NY, USA, 2004. ACM.

[8] S. Büttcher, C. Clarke, and I. Soboroff. The trec 2006 terabyte track. In *Proceedings of TREC 2006*, 2006.

[9] B. Carterette and J. Allan. Semiautomatic evaluation of retrieval systems using document similarities. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 873–876, New York, NY, USA, 2007. ACM.

[10] B. Carterette, J. Allan, and R. Sitaraman. Minimal test collections for retrieval evaluation. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 268–275, New York, NY, USA, 2006. ACM.

[11] C. Clarke, N. Craswell, and I. Soboroff. Overview of the trec 2004 terabyte track. In *Proceedings of TREC 2004*, 2004.

[12] C. Clarke, N. Craswell, and I. Soboroff. Overview of the trec 2009 web track. In *Notebook Proceedings of TREC 2009*, 2009.

[13] C. Clarke, F. Scholer, and I. Soboroff. The trec 2005 terabyte track. In *Proceedings of TREC 2005*, 2005.

[14] G. V. Cormack, C. R. Palmer, and C. L. A. Clarke. Efficient construction of large test collections, 1998.

[15] P. Doreian. Measuring the relative standing of disciplinary journals. *Inf. Process. Manage.*, 24(1):45–56, 1988.

[16] P. Doreian. A measure of standing for citation networks within a wider environment. *Inf. Process. Manage.*, 30(1):21–31, 1994.

[17] N. L. Geller. On the citation influence methodology of pinski and narin. *Information Processing and Management*, 14:93–95, 1978.

[18] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 576–587. VLDB Endowment, 2004.

[19] T. H. Haveliwala. Topic-sensitive pagerank. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 517–526, New York, NY, USA, 2002. ACM.

[20] G. Jeh and J. Widom. Scaling personalized web search. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 271–279, New York, NY, USA, 2003. ACM.

[21] E. Jensen, S. Beitzel, A. Chowdhury, and O. Freider. Repeatable evaluation of search services in dynamic environments. *ACM Transactions on Information Systems (ACM-TOIS)*, 26(1):1, November 2007.

[22] K. S. Jones and van Reijsbergen. Information retrieval test collections. *Journal of Documentation*, 32(1):59–75, 1976.

[23] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J.ACM*, 46(5):604–632, 1999.

[24] A. Özgür, L. Özgür, and T. Güngör. Text categorization with class-based and corpus-based keyword selection. In *ISCIS*, pages 606–615, 2005.

[25] G. Pinski and F. Narin. Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics. *Information Processing and Management*, 12:297–312, 1976.

[26] R. Sinha and R. Mihalcea. Unsupervised graph-basedword sense disambiguation using measures of word semantic similarity. In *ICSC '07: Proceedings of the International Conference on Semantic Computing*, pages 363–369, Washington, DC, USA, 2007. IEEE Computer Society.

[27] E. M. Voorhees and D. K. Harman. *TREC Experiment and Evaluation in Information Retrieval*. MIT-Press, September 2005.

[28] E. Yilmaz and J. A. Aslam. Estimating average precision with incomplete and imperfect judgments. In *CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 102–111, New York, NY, USA, 2006. ACM.