Context Driven Ranking for the Web

Siddharth Jonathan J.B., Andreas Paepcke, Hector Garcia-Molina
Stanford University

ABSTRACT

Users of Web Search Engines have traditionally been known to mostly restrict their explorations to the first few pages. Studies [2] have shown that only about 7% of users look beyond the first 3 pages of results. To make searching huge collections of data useful, it is very important that the results returned are ordered or ranked correctly. Ranking of Web pages today revolves around 3 categories of algorithms. First, there are algorithms that look at just the text of the document to decide how good a match it is for a certain query. Secondly, there is a class of algorithms that make use of link information. Thirdly there are numerous 'hacks' that yield surprisingly valuable cues as to how relevant a retrieved document is to a query. In this report I will be focusing on the first type of algorithms. In this sense, the approaches that I talk about can be extended to documents other than Web pages as well. I will also be presenting my evaluation framework and how my algorithms compared against a baseline system.

1. INTRODUCTION

For Web Search and any other form of search that involves large collections of data, ranking of results that are returned in response to a query is of utmost importance. It is this problem of ranking pages effectively that we address in this report.

One of the reasons why ranking of Web pages is a hard problem is because of the limited amount of information available at query time to score documents by. Typically, the user inputs a few key words and that is all the information the search engine obtains at query time. Speed at query time is also an important concern since response times need to be kept low. This precludes any complicated processing performed at query time.

2. EXISTING RANKING METHODS

Ranking of Web pages today revolves around 3 categories of algorithms. Firstly, there are algorithms that look at just the text of the document to decide how good a match it is for a certain query. The scheme that has worked best has been the *tf-idf* score. The *tf* or term frequency indicates how often a term appears in a document. The *idf* or inverse document frequency is a measure of the rarity of the term. It is generally taken to be the reciprocal of the number of documents in which the term appears. For example, the word 'the' would have a very low *idf* since it occurs commonly in most documents. It would have a high *tf* though.

Secondly, there is a class of algorithms that make use of link information. This includes algorithms like PageRank [4] and HITS [3]. Looking at information contributed by anchor text can also be considered to fall under this category.

Thirdly there are numerous 'hacks' that surprisingly yield valuable cues as to how relevant a retrieved document is to a query. These include for example the size of the font

with which the query term occurs in the document, the html tags with which it appears, the location of the query term in the page and countless others.

In this report, we focus on ways to improve the first category of algorithms. Our underlying hypothesis throughout all our experiments is that, in addition to looking at just the count (and maybe *idf*) of the keyword, it would be useful to consider the context in which that keyword appeared in a document. We would do this asynchronously and not at query time and we hope that this would address the issue of limited information available at query time to score documents by.

We came up with 3 algorithms that attempt to make use of *context* to rank pages. We evaluated these algorithms qualitatively and quantitatively. We noticed improvements over using just *tf-idf* scores. The improvements, though not significant on the average case, do justify our intuitions about the use of *context* as an additional metric to be used to rank pages.

3. RATIONALE BEHIND USAGE OF CONTEXT TO RANK WEBPAGES

In this section we describe with an example, how using *context* helps in the ranking of pages. Take for example, the query, *Motorcycl*e. The *tf-idf* based scheme would just score documents based on how often the term *Motorcycle* appeared. The benefits of this approach are that it is very simple to implement and very fast at query time. The drawbacks of this scheme are that it is blind to other words in the document like say *two wheeler*, *bike* and *oil* which might be good cues to score the document by given that the query term is *Motorcycle*.

For example, in the extreme case, when using the *tf-idf* scheme, a query for the term *Motorcycle* might return a document in which a person talks about his trip to a nearby city on his motorcycle. This document might not be the best result for a query on *Motorcycle* regardless of how many times the person mentions the word motorcycle in his document since the document is not really about motorcycles. The hypothesis is that someone who types in *Motorcycle* as a query presumably wants to know more about motorcycles and a document that talks *about* motorcycles would be a better result to return as compared to a document that mentions motorcycles multiple times but which is not really *about* motorcycles.

With our approach of trying to make use of the *context* in which terms appear, to rank pages, there is now the big question of how to identify this *context*. Ideally, we would like to automatically identify the context in which *motorcycle* appears most often, and use that to judge documents. If we somehow knew for example, that motorcycles generally appear in the context of *bikes*, *two wheelers*, *ride*, *oil* etc., it might help us weed away the outliers more effectively and rank documents better by making use of the presence of these context words. These context words give us more information about the query regardless of how brief the query is.

Our motivation in CDR was to implicitly deduce this *context* from the corpus and make use of this to rank pages more effectively. We also wanted to ensure that the algorithms we suggest, scale to large web collections and do not pose significant overhead at query time. We propose 3 algorithms namely, *GenMod*, *GenMod*++, and *CDR-Chi* that make use of *context* to rank pages. We explain these algorithms in detail in the sections below along with our evaluation of their effectiveness.

4. ALGORITHMS EXPLOITING CONTEXT

4.2 *GenMod*- BUILDING A GENERATIVE MODEL OUT OF THE POSTINGS LIST

4.2.1. *GenMod-* METHODOLOGY

The underlying hypothesis here is that all documents in a postings list (the set of all documents in which a query term appeared) of a web index are generated by an implicit generative model. Ranking the documents by the probability of generation by the model might be a good idea.

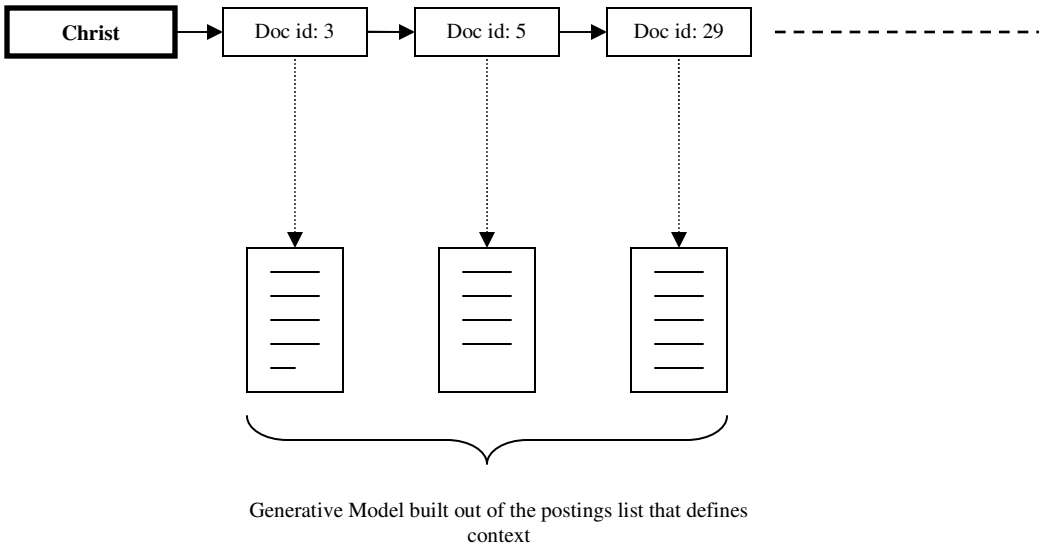

Figure 1. A view of one entry in an inverted index of a Web Search engine. The indexed keyword is *Christ*. The postings list shows that documents 3,5 and 29 contain that term. The postings list contains pointers shown by dotted lines to the actual document.

Using the postings list for a query term as a training set, we trained a multinomial Naïve Bayes Classifier. This was a single class classification problem in which the class that we were interested in was all documents that contained the query term in it. This class is effectively the postings list for that query term. After this training phase, we 'tested' on the same set of documents. During this testing phase, given a document from the postings list, the model outputs a score for it, which is the document's probability of generation given this multinomial model.

We then used these probabilities to rank the documents. To avoid penalizing long documents which would invariably have lower probabilities of generation since we are multiplying probabilities for tokens together, in the multinomial model, we normalized for length.

As an example for what we hope to see, consider this example. Consider the query *Christ*. Let us take 2 documents to be ranked or ordered, *x* and *y*. Document *x* has 5 words in all. Three of those are *Christ* while the remaining two are completely irrelevant. Document *y* has 5 words as well. Two of these are *Christ*. The remaining three words are 'Jesus', 'faith' and 'Bible'. The simple *tf-idf* based scheme outlined earlier will rank document *x* above document *y* which goes against our intuition. *GenMod* will rank *y* higher. This is

because in a training set consisting of documents that have *Christ* in them, words like 'Jesus', 'faith' and 'Bible' are more likely to have occurred i.e have a higher probability of generation compared to other random and completely irrelevant words like say 'budget' or 'physics'. This deduction of the context of *Christ* to comprise of words like 'bible', 'faith', 'Jesus' etc. was made by the classifier, when it went through the entire postings list for 'Christ' and counted word occurrences. Presumably, (after dealing with stop words) the classifier found these words (Jesus, bible etc.) to occur relatively more often in the corpus.

4.2.2. *GenMod* EVALUATION

To qualitatively evaluate GenMod, we indexed and trained on the Cranfield collection, a set of 1398 abstracts of aerodynamics journal articles. Sample queries were run on this corpus using both the *tf-idf* scheme and the GenMod schemes. The results of these queries were sent to domain experts in aerodynamics who judged which engine was better. The engines were referred to as engine X and engine Y and this naming was counterbalanced to prevent any bias. The judges were contacted by email and the results were received by email.

On analyzing the results, we found that the system did not seem to give results as satisfying as a simple *tf-idf* based system. In particular there was a lot of dissipation or lack of focus with respect to the query term in the results. It appeared that the system did not seem to give too much importance to the query term. The frequency of occurrence of the query term in the document was not given enough importance in the ranking. It was being handled just like any other term. The query term did not have the highest probability of generation as per the probabilistic model built by the classifier. This was because, even though the query term appeared in every document, it was clearly not the most frequent word. Therefore, the multinomial naïve bayes classifier did not give it the highest probability. The same reasoning applied to the *context* words. So clearly something needed to be done here.

Relevant related work:

People have used Language modeling for IR before. In "A General Language Model for IR", Song et al. talk about using the query as the object of generation(generative process) and build a language model for each document. The documents are then ranked by the probabilities of generation of the query [1]. They talk about various ways to deal with problems of data sparsity in their models. In my scheme I use the document (independent of the query) as the object of generation (generative process) and build the language model out of the entire postings list. Because my language model makes use of significantly more data, problems of data sparsity are less pronounced. My method also implies less processing at query time.

Furthermore, in their approaches since they look at just the query terms, the problems of not incorporating contextual information still remain.

GenMod++:

To overcome the problem of 'dissipation of focus' that was evident in *GenMod*, I realized that I had to give more importance to the query term and if possible to other words that were *context* words. I decided to give these words an artificially higher probability of generation by boosting their 'count' values rather than use the probability score derived from the corpus.

To decide which of the other words needed to be given more importance, I used the Chi-square metric which is a well known metric used for feature selection in classification problems. Here the classification problem is that of classifying documents into one of two categories. Those that belong to the postings list and those that do not. Using this metric, I came up with a 'boost set' of terms which corresponded quite accurately with the *context* of the query word. For example, for the query word 'Christ', the top scoring words as per the Chi-Square statistic were 'Jesus', 'faith', 'bible' etc. I artificially up-weighted these words so that the classifier gave them a higher probability of generation. I then separately boosted the importance of the query term even more since it was more important. All this was done so that a document which has a higher occurrence of words of the boost set and the keyword would be ranked higher than a document that did not. This was done to dispel the 'dissipation of focus' that appeared to have crept in to the algorithm.

GenMod++ Results:

Although this improved results to some extent, the results were still not satisfactory for all queries. There were some query words in which the results were comparable but on most others, a simple tf-idf scheme was still performing better.

One problem was the size of the boost set. It was not clear as to how big the boost set should be for each keyword since one size would definitely not fit all. It was also not clear how much to boost the *context* words by. Arbitrarily deciding on the size of the boost set and how much to boost those words by did not seem like the ideal way of doing things. To make things worse, the 'dissipation of focus' problem still reared its head with some queries.

CDR-Chi:

One interesting thing I noticed was that the Chi-Square scores that I was getting seemed to model the *context* pretty well. The keyword was getting the highest Chi-Square score and the other words that one would assume to define the *context* of the query word were getting high scores as well.

I then decided to use a scheme which completely did away with the generative model and used just the CHI-Square scores. To score a document in a postings list, I calculated the average Chi-Square score per term in the document. This ensured three things. Firstly, that I wasn't doing any processing at query time. Secondly, the stop words were also getting handled naturally, since stop words have very low Chi-Square scores. On the contrary, for both *GenMod* and *GenMod++,* a document that maybe had just the word 'the' repeated 3 times would be given a higher score than a document that mentioned the query word 3 times! Therefore, good stop word filtering is critical to the success of *GenMod* and *GenMod++*. Lastly, *CDR-Chi* also ensured that longer documents weren't unduly biased against.

Evaluation Framework:

I used the 20 News Groups dataset as my corpus to search across. This is a collection of about 20,000 USENET news group messages. This data was appropriate in that it was diverse in content and authorship allowing for search queries on topics as diverse as Microsoft Windows, Christianity, automobiles, politics etc.

CDR-Chi Results:

This scheme completely solved the 'dissipation of focus' problem and took *context* into account as well. To test my system against the tf-idf based system, I defined my evaluation framework as follows. Given the name of a news group as a query, I tried to determine, for a given number of results returned, '*k*', how many of the results belonged to that news group. I varied '*k*' to see how that changed the difference in performance between the two systems.

The result as shown by figure 1, was that both systems were comparable in performance on the average case. In one self-constructed example, I was able to show that my system performed better. My example was as follows. The query was 'Motorcycle'. I created a document that mentioned motorcycle just once, but instead used the term 'bike' (which is a synonym for motorcycle in many places) about 6 times. The *tf-idf* based system did not show this document as a relevant result in the top 10 results. *CDR-Chi* however showed it as the 3rd result. This by itself doesn't tell us much. I then changed the 'bike' terms to 'motorcycle'. Now the *tf-idf* based system showed it as the top result. This shows that tf-idf on the query term alone without looking at the context may miss out on cases like these.

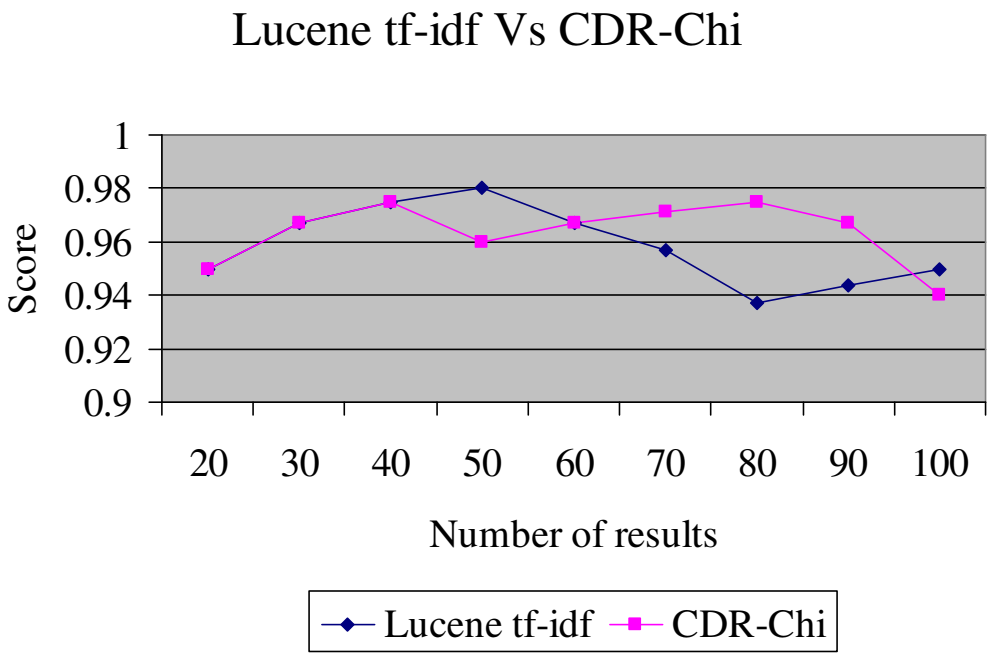

Figure 1: This chart shows the performance of *tf-idf* compared with the performance of *CDR-Chi*. The query term was the newsgroup 'motorcycle'. The score indicates the percentage of documents retrieved that actually belonged to the news group.

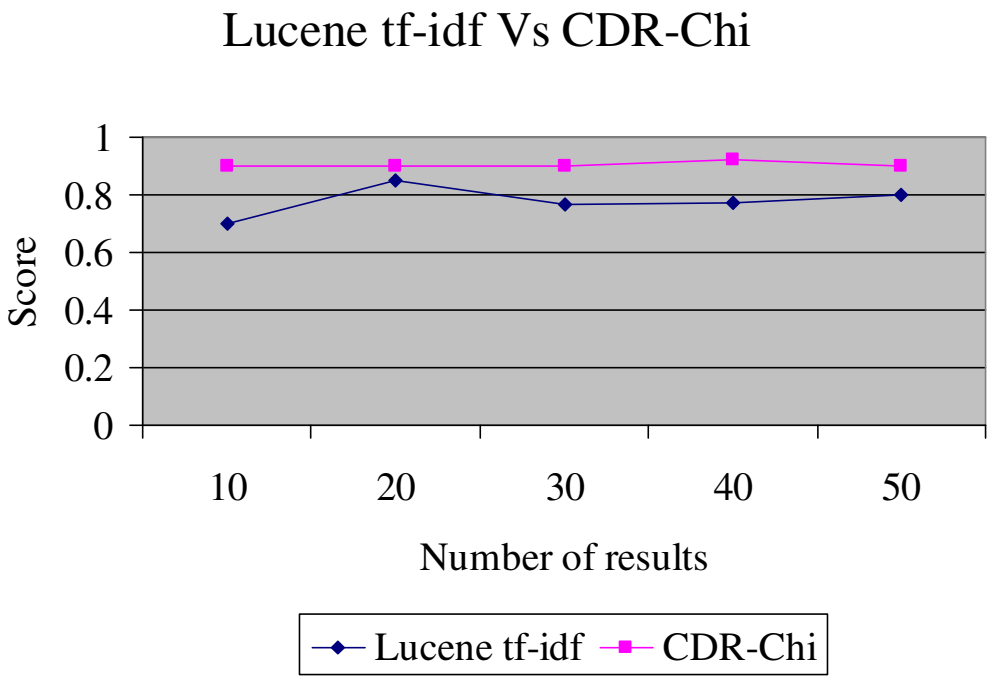

Figure 2: This chart shows the performance of *tf-idf* compared with the performance of *CDR-Chi*. The query term was the newsgroup 'medicine'. The score indicates the percentage of documents retrieved that actually belonged to the news group.

Although on a couple of cases *CDR-Chi* marginally outperformed *tf-idf*, on an average the performance of both systems were comparable in this test.

Intuition:

The results appeared to indicate that for this type of test, CDR-Chi marginally outperforms tf-idf when the query term has a high chance of belonging to more than one news group although semantically it is clear as to which unique news group it must belong to. Since CDR-Chi looks at context, it disambiguates better when ordering the results.

Conclusion:

Although these results indicate a coarse measure of performance, a real indicator for the best ranking of results would have to be a qualitative user evaluation. This is because this test does not account for the ordering of results. For example, if 10 results are returned for a particular query which was the name of a news group, 2 cases are possible. The first case is that the result set returned by *tf-idf* can be completely different from the result set returned by *CDR-Chi*. The second case is that the result set returned could be the same but re-ordered in *CDR-Chi*. Both these cases are treated the same way as per this test. A more detailed user study might bring out the difference between the two (if it exists).

References:

1. Ponte and Croft, A Language Modeling Approach to Information Retrieval.
2. First 20 precision among World Wide Web search services (search engines) H. Vernon Leighton , Jaideep Srivastava
3. J.M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," Proc. 9th ACM-SIAM Symp. Discrete Algorithms, ACM Press, New York and SIAM Press, Philadelphia, 1998, pp. 668-677.
4. S. Brin and L. Page, "The Anatomy of a Large Scale Hypertextual Web Search Engine," Proc. 7th World Wide Web Conf., Elsevier Science, Amsterdam, 1998, pp. 107-117.
5. Fei Song and W.Bruce Croft, "A General Language Model for Information Retrieval", Conference on Information and Knowledge Management archive Proceedings of the eighth international conference on Information and knowledge management table of contents Kansas City, Missouri, United States, Pages: 316 – 321, Year of Publication: 1999, ISBN:1-58113-146-1.