Mining Anchor Text for Query Refinement

Reiner Kraft and Jason Zien
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
{rekraft, jasonz}@almaden.ibm.com

ABSTRACT

When searching large hypertext document collections, it is often possible that there are too many results available for ambiguous queries. Query refinement is an interactive process of query modification that can be used to narrow down the scope of search results. We propose a new method for automatically generating refinements or related terms to queries by mining anchor text for a large hypertext document collection. We show that the usage of anchor text as a basis for query refinement produces high quality refinement suggestions that are significantly better in terms of perceived usefulness compared to refinements that are derived using the document content. Furthermore, our study suggests that anchor text refinements can also be used to augment traditional query refinement algorithms based on query logs, since they typically differ in coverage and produce different refinements. Our results are based on experiments on an anchor text collection of a large corporate intranet.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Query Formulation, Search Process*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

General Terms

Algorithms

Keywords

anchor text, Web search, query refinement, rank aggregation, query logs, searching and ranking, data mining

1. INTRODUCTION

The popularity of search engines on the Internet has led to a nearly uniform interface for searching: a single input box that accepts keywords. This simple interface has pushed the burden of inferring the intent of the user's information need down into the search engine. One of the most important classes of queries that must be answered are ambiguous queries - broad queries that have many results in possibly many different domains of knowledge, for instance, "research" or "java". In fact, short single word queries are quite commonly used [23]. Over 22.5% of web queries studied in [26] were single term queries. Because of this, there are potentially many matches, most of which are not relevant to the user's actual information need. The specific form of query refinement that we examine is one where the user's query is expanded with additional terms using an algorithm that automatically tailors its results to the information in the document collection.

A recent study [13] examined several aspects of anchor text (e.g., their relationship to titles, the frequency of queries that can be satisfied by anchor text alone) in a large intranet. They showed evidence that anchor text summaries, on a statistical basis at least, look very much like real user queries. This is because most anchor texts are succinct descriptions of the destination page. This similarity between search queries and anchor texts can be exploited with an algorithm that automatically completes, refines, or shows related queries, and helps users find relevant search results. Furthermore, the refinements help to better explore a knowledge domain in more depth in cases where a user has only partial domain knowledge.

The paper first defines the terminology used and describes the proposed anchor text mining techniques in more detail. Our approach is experimental, based on a study of a large corporate intranet comprising a document corpus of 4 million unique HTML documents.

We discuss the problem itself, possible solutions to it, and our solution to the problem, including details on how we collect and sort anchor text, preprocess and clean it to remove noise, perform quality analysis to obtain a static rank using rank aggregation methodologies, and develop tooling so that it can be integrated into a Web search engine to produce query refinements during query execution.

Then we explain how we built and used a quality benchmark, which allowed us to further tune the presented mining algorithms and experiment with different preprocessing, filtering steps, and relevancy ranking methods.

We then describe how we conducted a user study to evaluate the usefulness and usability of the proposed approach. The goal of this user study was to qualitatively compare our query refinement algorithms based on anchor text to techniques that are based solely on text obtained from a document corpus or a query log. We discuss the results obtained from the user study in detail.

At the end we compare the proposed methods with related work, and discuss future work.

We show that for the particular application of providing query refinements or suggesting related queries, our algorithms based on mining anchor text outperformed algorithms based on mining the document corpus. Furthermore, our anchor text refinements can also be used to augment traditional query refinement algorithms based on query logs, since they typically differ in coverage and produce different refinements.

The paper focuses on query refinement on a large corporate intranet, but we are confident that the proposed mining methodology for anchor text mining represents a general framework with other applications to Web search.

The main contribution of this paper is a novel use of anchor text for query refinement and its integration into a search engine so that it can be efficiently used during query execution.

We hope to stimulate more research to further enhance these mining algorithms, and build new applications that can help to improve the overall Web search experience and quality.

2. TERMS AND DEFINITIONS

First, we need to define what **anchor text** is. We define anchor text to be the "underlined or highlighted clickable text" that is displayed for a hyperlink in an HTML page when rendered in a Web browser (the text that appears within the bounds of an `<a></a>` HTML tag). For instance, for a tag of the form:

```
<a href="index.html">foo</a>
```

we would say that the anchor text is "foo", which is associated with the document "index.html".

All of the anchor text in a document corpus pointing to a target document is referred to as **anchor text document** for that document. So for each document with hyperlinks pointing to it (e.g., "index.html") we have an *"about"* document or simply *anchor text document* that comprises all anchor texts for that target.

We also use the synonym **anchor text summary** that refers to anchor text. However, an anchor text summary may also include some text before or after the anchor text to better capture the context of the anchor text.

3. REFINING QUERIES USING ANCHOR TEXT

Based on the observation by Eiron and McCurley [13] anchor texts and queries are very similar. A naive algorithm for query refinement would therefore just need to look at the original user query, find all anchor texts that are similar, and present these to the user as query refinements or related queries.

3.1 Problems with using raw anchor text

However, there are many difficulties that make this simple algorithm fail quickly: For popular queries, there are typically too many anchor texts that match a single term query. Furthermore, a large portion of anchor texts are automatically generated by Web authoring tools, or are for other reasons useless. We need to rank the importance of anchor text summaries according to the user's original search query to obtain a small list of high quality anchor texts that are similar and semantically related to the query, but should contain additional useful query terms. The anchor text summaries from this ranked list itself can then be used as subsequent queries.

One approach to the problem would be to cast this problem as a traditional document retrieval problem. We could index the anchor texts, treating each unique anchor text as a document, and then rank and return the documents using a normal search engine's $TF * IDF$ scoring. This approach has a number of limitations:

1. Treating each unique anchor text as a document does not take into account the multiplicity of that anchor text.

2. Documents with high TF (term occurrence count in a document) have no bearing on the quality of the refinement.

3. Repetitions of a term in a document may produce unnatural queries for refinement.

4. The IDF (inverted document frequency) is not relevant, since a large fraction of the queries we wish to refine are single term queries.

3.2 Using rank aggregation methods to select query refinements

Instead, we need a ranking method which can take multiple factors into account, and can incorporate the factors that are relevant to producing good query refinements. In fact, there is recent work [14] which shows that median rank aggregation is an efficient, flexible ranking algorithm for combining multiple factors which provides nice theoretical properties.

In median rank aggregation, given n elements (in our case, an element is a unique anchor text), and k lists of cost values over those elements, a final ranking is produced. To produce the final ranking, each list, L_i $(1 \leq i \leq k)$ is sorted to obtain a ranked ordering of the elements in each list. The final rank is generated by computing the median of the ranks of each element. We then sort the elements using that median value to obtain the final rank of the element (we break ties arbitrarily). Obviously, for large lists, this would be prohibitively expensive to compute at query time. Our system pre-computes the ranks of all of our query refinements once at index build time so that they are available at almost no cost at query execution time.

In our method, we used median rank aggregation over three factors:

1. The weighted number of occurrences of an anchor text $WCOUNT$. Our weighting was based on the particular type of anchor text (if the anchor was pointing to something in the same directory, on the same site, or a different site).

2. The number of terms in the anchor text. We would like query refinements to have few terms, since we wish to only slightly narrow the scope of a query.

3. The number of characters in the anchor text. We wished to have short, concise query refinements.

3.3 Benefits of anchor text

By making use of anchor text instead of generating refinements using document contents, we obtain a number of significant benefits:

- Anchor text data is typically an order of magnitude less than the total document collection data. Thus, processing it is much faster than processing the document collection.

- Pages with a large amount of anchor text pointing to it tend to have higher indegree (number of links pointing to it) and higher ranks based on link analysis. Thus, using these anchors as refinements leads to results that are relevant to the document collection.

- The use of anchor text summaries as candidates for refinement has another potential benefit. Since these refinements occur frequently in anchors, it is likely that these point to popular sites. Thus, the refinements capture a view of the popularly linked-to information in the data set. In addition, selecting one of the refinements then causes the search engine to retrieve results that contain these frequently occurring anchors, leading to good results.

3.4 Incorporating Refinements into a Search Engine

We integrated query refinements into the *Trevi* intranet search engine. Trevi is used to search the IBM intranet and serves all IBM employees worldwide. In Trevi, a query is sent down two paths. The primary path goes to the main index, where documents are ranked using a $TF * IDF$ scoring function combined with other factors such as lexical affinities and static ranks.

The other path is a mechanism called *"QuickLinks"* where a set of result values is returned based on a key lookup. The user's parsed query is looked up using an exact match on the quicklink keys and the results are returned in the order given by the quicklink phrase's pre-assigned static rank. This mechanism was designed to allow administrators to hardwire known good results to common or important queries for display at the top of the search results. Furthermore, this mechanism can also be used to search other data that conforms to a file-based key/value specification format.

When implementing our interactive tool for query refinements we decided to use this QuickLinks mechanism, since it represents a convenient interface for prototyping and experimenting with the refinement algorithms. Our main task essentially in implementing our anchor text query refinement method was in generating a list of index terms and phrases associated with each anchor text summary and associating a static rank to it. The way we mapped refinements into this framework was as follows: For each anchor text summary, we used a sliding window of size from one to the number of terms minus one in the summary, and we output each anchor phrase as a quicklink key. For instance, suppose we had the anchor: *"IBM Almaden Research Center"*. This would generate the following quicklink keys:

- IBM
- Almaden
- Research
- Center
- IBM Almaden
- Almaden Research
- Research Center
- IBM Almaden Research
- Almaden Research Center

Any query that contains one of the above quicklink keys would cause the anchor text summary *"IBM Almaden Research Center"* to be generated as a potential query refinement. At query time, the top $k = 5$ ranked quicklinks with a key matching the query are then returned and displayed to the user.

4. MINING ANCHOR TEXT

4.1 Data Set

Our crawl of the IBM intranet consisted of over 33 million anchor texts (2.8 GB of parsed anchor text data) and over 4.0 million unique documents (totaling 60GB of parsed document data) after duplicate elimination.

We plotted the distribution of the length of the the anchor texts (using the number of terms as the length) on the x axis and the number of occurrences of anchor texts having that length on the

y axis on a log-log scale in Figure 1, and, as expected, we see it is a Zipfian distribution. On that same graph, we also plotted the distributions for a query log of $349,471$ intranet searches, and it shows the same characteristics. The reason why it is shifted down is because the number of items in that log processed is two orders of magnitude less than the number of anchor texts. This anchor text data was used as the basis for all our experiments and user studies.
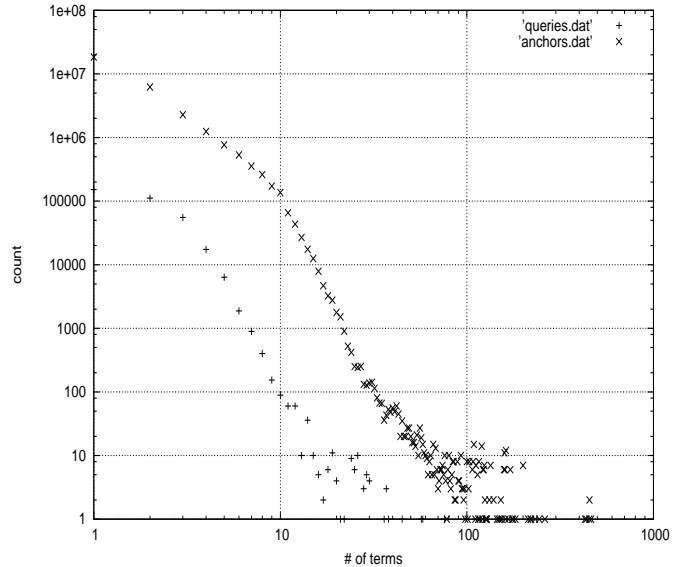

Figure 1: Histogram over our anchor text collection and a query log

4.2 Benchmark

To validate our approach we first built a benchmark consisting of 29 broad queries and manually rated roughly 5000 query refinements that were derived from our anchor text mining algorithms. See Appendix A for the list of these queries. Given a query, the rating was whether a given query refinement in our opinion was considered to be useful in refining the query or not. We then used this benchmark as a basis to further tune our anchor text mining algorithms. As a quality metric we chose to measure precision at five (P@5), because we want to measure the quality of the top five refinement suggestions displayed to the user.

4.3 Preprocessing anchor text summaries

A query refinement with too many terms might be too restrictive and too specific. We sought to determine the number of anchor terms needed to produce good query refinements by varying the number of terms allowed in anchor text summaries. We therefore introduced two parameters $MINCOUNT$ and $MAXCOUNT$. Anchor texts with too few (less than $MINCOUNT$) or too many terms (greater than $MAXCOUNT$) were discarded. We typically set $MINCOUNT$ to be two since refinements with smaller size are not useful in our scenario. We experimented then with limiting the maximal number of terms between $2 \leq MAXCOUNT \leq 5$. In the experiments when using our benchmark it turned out that $MAXCOUNT = 3$ produced the best refinements.

We define the term *window* to be a number of consecutive terms. When we say we are using a *window size* of k this means we are looking at k consecutive terms.

The *window size* was used as a parameter for our algorithms. For instance, we investigated the effect of counting stop words (see

Appendix B for the list) or not for choosing good refinements by considering the window size parameter. For instance, suppose we choose a window of size two. If we count stop words, the anchor text "the research center" has a count of three terms. If we are not counting stop words, it would have a count of two terms (since "the" is considered to be a stop word). Using the *count* and the *window size*, we determine if a anchor text refinement would pass the filter (e.g., $count \geq MINCOUNT$ and $count \leq MAXCOUNT$.

We need to point out that we did not eliminate anchor texts that contained stop words from the anchor text collection. Instead we relied on the window size to eliminate useless anchors. For instance, consider $MINCOUNT = 2$, and an anchor text "the solution". If we do not count stop words the number of counted terms for this anchor text would be one (since we ignore the term "the"), and therefore it would not pass the filter. However, the anchor text "cats and dogs" would have a count of two "(and" is stop word), and therefore it would pass the filter. The advantage of this method is that we keep the original annotation of the user who authored the anchor text, which leads to more natural-looking queries. If we would eliminate stop words the anchor text in the previous example would be converted to "cats dogs", which does not reflect the original intention of the author.

The best results were found with a window size of three, as shown in Figure 2. In addition, our experiments showed that counting stop words (the .SW curves) was generally slightly inferior to not counting them, though the results are very close in many cases.
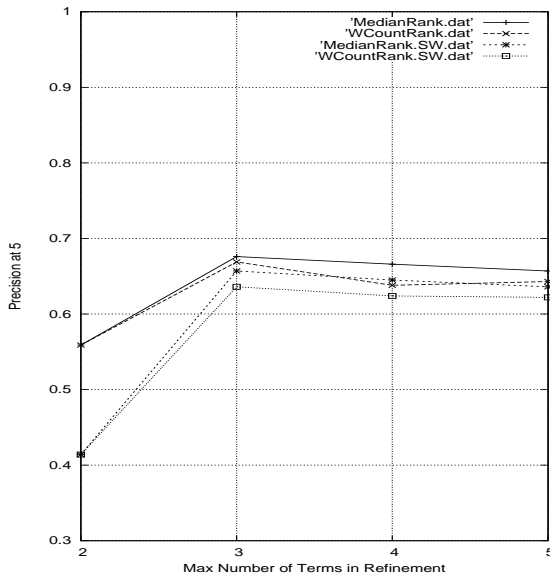

Figure 2: Precision @5 using various anchor text mining algorithms

4.4 Ranking of anchor text summaries

We compared two methods for generating the static ranks. In the $WCOUNT$ ranking style, we assigned a rank to each anchor text based on the following formula:

$$WCOUNT = UINT\_MAX - \\ (anchorCount * 1000000 \\ + anchorSameHostCount * 1000 \\ + anchorSameDirCount)$$

Each of the values, anchorCount, anchorSameHostCount, and anchorSameDirCount were limited to maximum values of 4000, 999, and 999 respectively in order to fit them into $WCOUNT$, which was represented as a 32-bit number ($UINT\_MAX$ is the maximum unsigned integer value for this 32-bit number)

In the median rank aggregation style, we use several ranking criteria:

1. the $WCOUNT$ (lower is better)

2. the number of terms in the anchor text summary (lower is better)

3. the number of characters in the anchor text summary (lower is better)

Each of these criteria essentially represents a cost function, and an anchor text will probably rank differently for each of these. For instance, an anchor text could have a low $WCOUNT$ cost, but too many terms. Rank aggregation will combine these different costs for each anchor text by calculating the median rank, and assigning this rank to be the static rank for this anchor text. With such a flexible scheme, one can easily incorporate other cost functions to influence the results.

We calculate the cost of each anchor text with all cost functions. This results in three lists of anchor texts, each of these lists ranked according to the corresponding cost function. Then we sort each list by its cost and compute the median rank based on the ordinal rank of those sorted factors for each anchor text summary to produce its final rank.

On empirical benchmarks we found that the median rank produced the best results, shown in Figure 2. In addition, we note that those methods which counted stopwords (the curves labelled with names ending in .SW) performed worse than those that did not, although for three or more maximum terms, they were all extremely close.

4.5 Performance

Since we generate all of the possible refinements at index build time, the large sorts needed to produce the median rank aggregation values have no impact on the query runtime. Generating all of the quicklinks for our anchor text took about 12 minutes on a 2.4Ghz IBM X345 server, so it was not prohibitively expensive.

The performance impact of the quicklinks feature on queries is negligible (in the tens of milliseconds), because typically there are very few quicklinks retrieved for each query, and there is usually only one disk access needed per query.

5. USER STUDY I: VALIDATION OF EVALUATION BENCHMARK

To validate the usefulness and quality of our algorithms, we conducted two user studies. The focus of the first study was to compare our query refinement algorithms based on anchor text mining to two similar mining methods that are based on the document collection text, and measure the qualitative differences using the queries from our quality benchmark. Then in the second user study we use different queries to see how the algorithms generalize and also include query log data for comparison.

Overall we had 37 users participating in user study I. Each user had to evaluate the usefulness of query refinements for a list of twenty queries. For each query we presented up to 25 query refinements. Overall we collected 8,904 user ratings.

5.1 Algorithms

The purpose of the first user study was to compare the quality of query refinement suggestions of three algorithms:

1. ANCHOR We picked the anchor text refinement algorithm that was the best from our quality benchmark. For further reference we will refer to this refinement algorithm as ANCHOR. The ANCHOR method used median rank aggregation described in Section 4.4 and did not count stop words.

2. DOC.SW The first document collection based method which we wanted to compare against, which we will refer to as DOC.SW, gathers all of the most frequently occurring two and three term phrases (using the number of documents they occur in, not their total occurrences). The phrases were gathered by counting occurrences of all phrases using a sliding window through the full text of all of the documents in the collection. We then ranked these phrases using a median rank based on the document occurrence count (higher being better), the number of terms in the phrase (fewer being better), and the number of characters in the phrase (fewer being better).

3. DOC The second document collection based method we compared to was based on DOC.SW, except that stop words were not counted when evaluating a phrase's word count (so, for instance, "the thinkpad" would not be considered since it did not contain two or three valid words). We will refer to this method as DOC.

As a corpus we used the same large document collection that was used to tune the mining algorithms comprising roughly 4 million unique HTML documents collected from the IBM intranet. Since the total document data set is an order of magnitude larger than the anchor text data, it is not surprising that one of the disadvantages of the DOC and DOC.SW methods is the substantial processing time needed to produce the query refinements.

Since we wanted to focus on the effect of the used document corpus we made sure that the basic mining algorithm used for ANCHOR, DOC.SW, and DOC.SW was as similar as possible to allow for a fair comparison.

5.2 Conducting the User Study

We picked twenty broad topics (a subset of our benchmark topic set). See Appendix C for the complete list.

For each of these topics we generated five query refinements per algorithm: We picked the five top ranked suggestions from ANCHOR, DOC.SW, and DOC and mixed them randomly into one list. Duplicate refinements were collapsed into one. We then presented the broad topic along with the list of refinements, and asked the users to rate the usefulness of these refinements.

The instructions we presented to the users before were as follows:

Over the next pages we will show you various search queries or topics. Imagine that for a specific topic you want to learn more about it and explore what is out there on the IBM intranet. We show you a list of potentially related queries for each topic. We ask you then to

rate each of these related search queries whether you think it is helpful in learning more about the original topic or not. You can ask yourself the question "If I want to learn more about that topic, would I issue the presented query to a search engine in hope of learning more about that topic? Or, do I think that the presented query is useless to further explore the topic?" Also, does the query suggestion help to refine or narrow down the scope of a broad topic?

We then asked the users to rate each refinement as

useful or helpful to explore, learn more about, or refine the topic on the IBM intranet.

not useful or helpful to explore, learn more about, or refine the topic on the IBM intranet.

don't know in case the user was uncertain.

and provided an example for each of these ratings.

We present one example for the topic "java" to better illustrate what the generated refinements look like:

- java client
- java education
- v8 java
- the java
- tool java
- java 1
- developer java
- visualage for java
- using java
- java xml
- for java
- regex for java

5.3 Results

Rating	ANCHOR	DOC	DOC.SW
Useful	0.76	0.55	0.36
Not Useful	0.17	0.36	0.58
Don't Know	0.07	0.09	0.06

Figure 3: Fraction of votes for each method

The user study (Figure 3) conclusively confirmed that our mining techniques for anchor texts outperform similar techniques that are based on document texts in the quality of the suggested refinements: The precision at five for the ANCHOR method was 0.76, which was 38% better than the DOC method, and 111% better than the DOC.SW method.

Manually examining the refinements generated by DOC.SW, we found that it suffered from poor precision because it often included stop words and prepositions in its refinements. The DOC method was much better, because stop words were not counted. However, this method often generated refinements, which were unnatural, or that were not general concepts. In contrast, the ANCHOR method was able to produce excellent refinements which expressed the concepts reflected by the corpus.

6. USER STUDY II

Since User Study I used queries from our evaluation benchmark, one question which comes to mind is, was our method tuned and overspecialized to our training set? User Study II performs exactly the same study as User Study I, however, we instead used a different set of 22 queries to show that our method generalizes to queries not in our training set.

In addition, we added another data set for query refinement for comparison: query logs. Based on the similarity between anchor text and search queries we expected query logs to be quite valuable as an additional source to possibly augment anchor text data.

We processed and ranked queries in the query logs in a manner as similar to our anchor text algorithm as possible, to see if real user queries would be useful as queries for refinement. Our query log contained 349,471 real user queries that were collected over a four month period. The processing steps comprised using the median rank of their frequency (higher is better), number of terms (lower is better), and number of characters (lower is better). We refer to this method as QUERYLOG. In QUERYLOG stop words were not counted. In addition, we introduced the QUERYLOG.SW method where stop words were counted.

	Useful	Not Useful	Don't Know
ANCHOR.SW	0.64	0.25	0.11
ANCHOR	0.63	0.27	0.10
QUERYLOG	0.63	0.24	0.12
QUERYLOG.SW	0.63	0.25	0.12
DOC	0.51	0.40	0.09
DOC.SW	0.30	0.64	0.06

Figure 4: Fraction of votes for each method

User Study II received 13,177 total ratings from 30 unique participants. From Figure 4, we see that the anchor text-based methods and query log-based methods were nearly identical. Both sources of data produced queries that were judged useful for refinement by users. Since the query log data consists of actual user queries, we expected that users in the study would judge them as good candidates for refinement, so we were pleased that our anchor-based method generated candidates were just as good or better. Does that mean that one of the data sources (anchors or query logs) is irrelevant?

We examined the overlap of anchor texts and query logs to answer this question. If there is a significant overlap, then one of the data sources would be redundant. We first processed the query logs, and broke them down into two halves ($QL1$ and $QL2$). In order to get a baseline number, we compared the overlap between $QL1$ and $QL2$. We took all unique queries with at least two terms, and found that $QL1$ had 47,405 queries and $QL2$ had 47,915 queries, and there were 6,577 that were common to both. Next, we gathered all of the unique anchor texts with at least two terms (there were 457,288 of them), and found that there were only 2,180 that overlapped with $QL1$. Similarly, when comparing to $QL2$, the anchor texts overlapped only in 2,287 queries. In both cases, this is substantially fewer than the overlap between $QL1$ and $QL2$.

We conclude that the coverage of anchor texts and query logs differs substantially enough that both are useful for query refinements. However, as we have pointed out earlier, anchor texts have advantages in that they are derived from the actual data, and thus reflect the document collection's actual content. In contrast, query logs reflect the exploration of users. In some cases they may reflect the data set, in other cases, they may just reflect the users' hopes about what the data set contains.

In summary, our anchor text based methods have produced good query refinements, substantially outperforming document-based methods. In addition, we found that using our median rank aggregation method over query logs perform equally well, though the coverage differs, leading to different refinements that can be used to augment anchor text based refinements.

7. RELATED WORK

Related work can be broken down into three broad categories: First, work related to anchor text processing. Second, ranking and rank aggregation methodologies for Web search. Third, research related to query refinement.

7.1 Anchor Text Processing

The idea of propagating anchor text to the page it refers to was published and implemented by McBryan [20] in 1994. Later Brin and Page [3] implemented the first prototype of the Google search engine. They pointed out that one of the useful properties of anchor text is that it often provides more accurate descriptions of web pages than the pages themselves. Second, anchor text exists for pages or content that could not be otherwise indexed by a text search engine (e.g., images). In these scenarios anchor text was mostly used to improve indexing and retrieval of a hypertext document collection. Our proposed mining and ranking algorithms take anchor text processing to a higher level. The proposed rank aggregation methods can use application specific ranks, which make them possibly suitable for a variety of Web search related tools applications (e.g., spell checking).

In [9] ARC (Automatic Resource Compiler) was developed as part of the CLEVER project to automatically generate lists of hubs and authorities related to ambiguous queries. They used the anchor text as well as a window of terms around the anchor text to determine if a destination was on topic or not, and adjusted the weights of the links in their web graph accordingly.

Other applications of anchor text are in the area of cross-language information retrieval where mining of anchor texts and link structures is used for automatically extracting translations of Web query terms [19].

Overall our mining algorithms are more general: We first pre-process all anchor texts to remove noise and low-quality content. Then we perform a ranking of this anchor text collection using rank aggregation methodologies. Depending on the application, we can plug-in different cost functions. Furthermore, the algorithms take various parameters that can be adjusted to specific needs. That results in an application specific tuned anchor text collection, which can then be used for different purposes. It would be interesting to investigate how cross-language information retrieval would benefit from our mined anchor text collection.

Craswell and Hawking [11] studied anchor text in the context of finding specific Web sites (the site finding task). Queries of this nature are also referred to as *navigational queries* [4]) or home page finding queries. In their experiments they did a comparison between using a document collection versus using an anchor text collection. The retrieval methods and ranking algorithms used in both cases were exactly the same. In their results they pointed out a strong advantage of using the anchor text based method to find specific Web sites. They concluded that anchor information is more useful than content for the site finding task, even if it is not further pre-processed or tuned. Our experiments also suggests that an anchor text collection is more useful for query refinement than a document collection. Furthermore, they point out another strength of the anchor text collection: Anchor text can enable retrieval with common misspellings or alternate names. This property enables

our refinement algorithms to suggest refinements when traditional content based refinement techniques would fail: Our refinements are based on the collective input of possibly many different authors expressing the same topic in a variety of styles and different wordings (including errors and typos).

Furthermore, Eiron and McCurley [13] provide very interesting insights and findings related to anchor text. They argue that anchor text is typically very short, and provides a summarization of the target document in context of the source document. They studied a search query log and confirmed that search queries are typically only a few words long, and express a summary of a subject that the user is interested in. One of their key observations is that (on a statistical basis at least) anchor texts and real user search queries are very similar. These observations initially motivated us to investigate whether we can leverage this similarity between anchor text and search queries and use it as a basis for query refinement. However, we realized that the proposed mining techniques can be applied in a more general context, and may be used possibly for various other Web search tools and applications.

7.2 Rank Aggregation Methodologies

There are a plethora of different ranking methodologies for Web information retrieval available [16], [17]. We can break them down into two categories: *static* and *dynamic* schemes. A static cost represents a global quality assessment of a document based on some criteria independent of the search query, whereas a dynamic cost (e.g., TF*IDF) is calculated during query execution and takes the search query itself into consideration. For instance, a popular static ranking cost function is Pagerank [21], which is used by major Web search engines. In between there are hybrid approaches that combine static and dynamic ranks (e.g., GURU [5]).

The problem we were facing after pre-processing the anchor text collection was that many terms would appear often in different anchor texts. Given a broad query term this will lead to an unmanageable number of refinements suggestion for a user. It was clear that we need to have at least one scoring method that evaluates the quality of an anchor text for being a "good" refinement. Since there are many desirable properties for refinements (e.g., length, number of terms) we required a method that systematically combines different static cost functions in a fair way.

Rank aggregation methods as proposed by Dwork et al. [12] address exactly this problem. These methods are based on social choice theory. They are typically computationally expensive and are targeted at small lists. In our work we also need to rank and prioritize large lists of anchor texts. In addition, we may have many different static ranks as described earlier, which makes it difficult to apply these techniques. However, more recent rank aggregation work [14] addresses this scaling problem. Therefore we adopted a similar static rank aggregation using the median rank for large lists to obtain one static rank per anchor text. This rank is then used to prioritize anchor text during query processing.

Overall the topic of ranking is highly subjective. There may be other static factors or features for query refinement that we have not considered yet. We have chosen the rank aggregation architecture, because it is quite flexible and extensible: It makes it easy to experiment with different scoring functions (e.g. add, remove, or replace a new scoring function).

7.3 Query Refinement

Query refinement is a well-known and important information retrieval tool (see Kobayashi and Takeda [17] for a survey on Web information retrieval).

Query refinement comprises query modification or expansion techniques [7], [22] as well as relevance feedback [6]. We briefly describe related work and how it differs from our approach. For the purpose of our paper we are mostly concerned with work related to query expansion or query refinement in general.

Besides the application of relevance feedback, query expansion or modification is typically based on either local [25] (i.e., results sets) or global [16] (i.e., thesauri) document analysis. Instead, our work is focused on an anchor text collection derived from a hypertext document collection (e.g., HTML corpus). Looking at anchor text, instead of using local or global document analysis represents a major contribution of this paper.

A recent example where a major search engine started to incorporate query refinement in its search application is AltaVista's Prisma^{TM} tool [1]. This tool enables interactive narrowing of search result sets. Anick analyzed users' session logs to investigate their Web search behavior when offered a simple form of interactive query refinement using Prisma^{TM}. We implemented a similar interactive query refinement tool that is based on our anchor text mining algorithm and integrated it into our Trevi intranet search engine. Our motivation was to collect usage data on how users are accepting the tool. Over a period of roughly three months we collected 119,042 queries. Only 1,707 of them were derived from our interactive refinement tool ($\sim 1.4\%$). This confirms Anick's results where also the vast majority of reformulations were still done manually. We intend to conduct more research on how to further improve interactive tools to increase the tool usage.

Carmel et al. [8] proposed a method of using lexical affinities in which lexical affinities were used to automatically refine queries. Their work differs from ours in that they do not show the users the lexical affinities, but instead, use them to reorder the results returned, and also, they do not make special use of anchor text.

Cooper and Byrd [10] developed *OBIWAN*, a Java based system, which is able to recognize domain-specific multi-word names and terms. This system comprises a *Context Thesaurus*, which allows users to look up vocabulary items that are related to the original query terms. Again, the difference to our method is that we have chosen anchor text as a basis to produce refinement suggestions, whereas OBIWAN uses *pseudo-documents* that are derived from the original document collection. For each vocabulary item there is one pseudo document, and it contains context information in which the item occurs in the collection. Similarly, our anchor text comprises context information for the query terms. There are different approaches to query refinement [2] and finding related topic terms [18] but all share in common that they are based on a document corpus. Our user study showed that the quality of our anchor text refinement suggestions outperforms document based suggestions.

Vélez et al. [24] introduce the notion of *concept recall*, an experimental measure of an algorithm's ability to suggest terms humans have judged to be semantically related to an information need. We set up our user study to achieve something similar to concept recall: We asked whether a refinement suggestion is *useful, or helpful to explore, learn more about, or refine a broad topic*. The emphasis really was on validating that the algorithms produce good and useful query refinements, which they did.

In addition, Vélez et al. measure the precision improvement of their refinement algorithms. The idea is to measure the effect of adding suggested terms to the original query. In our experiments we manually investigated the precision (P@10) of the original query, and some queries with added query terms that were suggested by our algorithm. We rated each search result item either to be good or useful, or not useful and compared the precision. However, we realized quickly that both the original query, as well as refinement queries produced good results. It all really depends on the infor-

mation need of the user and what he/she deems to be relevant for a certain topic: For one broad topic there can be many different refined topics that are relevant.

We therefore have chosen to not measure precision improvement of the resulting documents from a refined query, but measure instead the precision of the list of suggested refinement terms (P@5). Our rationale is that the goal of query refinement should be to assist users in articulating the appropriate query to satisfy their information need with the assumption that the underlying search engine already returns highly relevant results for a given query. Therefore if our algorithms produce a good list of refinement terms that are useful to the user and capture their information need, we helped the user to articulate a good query that satisfies their information need.

The usage of query logs as a basis for query refinement was studied by Fitzpatrick and Dent [15]. The advantage of using query logs as a basis for query refinement algorithms is that they are relatively easy to get even if there is no link information available. In general we think that query refinement that is using query logs represents a complementary approach that can be used to augment our anchor text based approach. The results obtained from the user study further support that query logs also represent a valuable data source that can help in the refinement process. One clear advantage of anchor text over query logs is one of bootstrapping: Obtaining large and useful query logs requires some initial effort where the time to obtain these depends on the search engine traffic. Whereas our anchor text refinements can be generated during indexing and are available to be used for refinement algorithms instantly after the index build process.

8. CONCLUSIONS AND FUTURE WORK

We devised a simple and elegant approach to automatic query refinement using anchor text and median rank aggregation. Our method and implementation worked remarkably well in practice. Our experiments and the conducted user studies which showed that the proposed mining techniques quantitatively outperformed similar methods using document collections for suggesting query refinements or related queries. Furthermore, by exploiting the similarity between anchor text and search queries the discussed algorithms provide high quality refinements for one and two term queries, which represent most of typical Web search engine's traffic.

There are more potentially interesting applications (e.g., spell checking) that can leverage from the proposed anchor text mining techniques. We want to investigate in future work how the proposed mining and tuning methods for anchor text can be used and integrated into a spell checker for Web search.

Another interesting aspect is the usage of lexical affinities which we have not investigated yet. Can LAs be used to further tune and improve the anchor text mining process?

Instead of narrowing a broad query, we can use Query refinement also to broaden a query with only a few results. This is often useful if the user types in a very restricted query and retrieves only one or two results. We can envision doing a similar technique for broadening queries using the proposed anchor text based refinement algorithm. Based on the number of returned search results the search engine could then pick which direction the refinement should work.

In our experiments we focused on anchor text, and did not include additional pre-anchor or post-anchor text, although it might be interesting to investigate on whether this would further improve our mining algorithms.

We hope that the paper stimulates interest to develop interesting applications for anchor text mining, and that Web search engines start integrating some of the ideas to further improve their overall search experience.

10. REFERENCES

[1] P. Anick. Using terminological feedback for web search refinement: a log-based study. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 88–95. ACM Press, 2003.

[2] P. G. Anick and S. Tipirneni. The paraphrase search assistant: terminological feedback for iterative information seeking. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 153–159. ACM Press, 1999.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[4] A. Z. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2), 2002.

[5] E. W. Brown and H. A. Chong. The GURU system in TREC-6. In *Text REtrieval Conference*, pages 535–540, 1997.

[6] C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*. Springer-Verlag, 1994.

[7] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using SMART: TREC 3. In *Text REtrieval Conference*, pages 69–80, 1994.

[8] D. Carmel, E. Farchi, Y. Petruschka, and A. Soffer. Automatic query refinement using lexical affinities with maximal information gain. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 283–290. ACM Press, 2002.

[9] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource compilation by analyzing hyperlink structure and associated text. *Proceedings of the 7th World Wide Web Conference*, 1998.

[10] J. Cooper and R. Byrd. OBIWAN a visual interface for prompted query refinement. *H1CSS31, Hawaii, USA*, 2:277–285, January 1998.

[11] N. Craswell, D. Hawking, and S. Robertson. Effective site finding using link anchor information. In *Research and Development in Information Retrieval*, pages 250–257, 2001.

[12] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the Tenth International Conference on World Wide Web*, pages 613–622. ACM Press, 2001.

[13] N. Eiron and K. S. McCurley. Analysis of anchor text for web search. In *Proceedings of the 26th Annual International*

ACM SIGIR Conference on Research and Development in Information Retrieval, pages 459–460. ACM Press, 2003.

[14] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 301–312. ACM Press, 2003.

[15] L. Fitzpatrick and M. Dent. Automatic feedback using past queries: social searching? In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 306–313. ACM Press, 1997.

[16] W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures & Algorithms*. Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[17] M. Kobayashi and K. Takeda. Information retrieval on the web. *ACM Comput. Surv.*, 32(2):144–173, 2000.

[18] D. Lawrie, W. B. Croft, and A. Rosenberg. Finding topic words for hierarchical summarization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 349–357. ACM Press, 2001.

[19] W.-H. Lu, L.-F. Chien, and H.-J. Lee. Translation of web queries using anchor text mining. *ACM Transactions on Asian Language Information Processing (TALIP)*, 1(2):159–172, 2002.

[20] O. A. McBryan. GENVL and WWWW: Tools for taming the web. *In World Wide Web Conference (WWW'94), Geneva, Switzerland*, 1994.

[21] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[22] Y. Qiu and H.-P. Frei. Concept-based query expansion. In *Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval*, pages 160–169, Pittsburgh, US, 1993.

[23] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[24] B. Vélez, R. Weiss, M. A. Sheldon, and D. K. Gifford. Fast and effective query refinement. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 6–15. ACM Press, 1997.

[25] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, 1996.

[26] J. Zien, J. Meyer, J. Tomlin, and J. Liu. Web query characteristics and their implications on search engines. *IBM Research Report*, RJ 10199, November 2000.

APPENDIX

A. BENCHMARK TOPICS

In our benchmark we used the following (broad) queries and topics:

almaden	hr
developerworks	research
health care	pension
travel	intranet password
db2	xml
linux	java
sales	storage
quantum computing	export regulations
open source	san francisco
lotus	lotus notes
alphaworks	thinkpad
human resources	global services
stock	websphere
stock options	travel reservations
knowledge management	

B. ANCHOR TEXT STOP WORDS

We used the following stop words for anchor text:

ibm	web	site	website	websites
link	next	topic	domain	prev
previous	page	to	the	for
and	of	an	or	not
a	click	here	–	&

C. USER STUDY TOPICS

These were the topics we used in the evaluation benchmark and user study I:

almaden	hr	alphaworks
thinkpad	pension	stock
db2	xml	linux
storage	lotus	rational
research	developerWorks	award
travel	human resources	technology
java	websphere	

These were the topics we used in user study II:

transcoding	mq	consulting
patents	retirement	security
portal	researchers	publishing
analyst	conference	odis
savings	websphere	hotel
tivoli	journal	drive
badge	learning	executive
library		