

Anchor Text Extraction for Academic Search

Shuming Shi¹ Fei Xing^{2*} Mingjie Zhu^{3*} Zaiqing Nie¹ Ji-Rong Wen¹

¹Microsoft Research Asia

²Alibaba Group, China

³University of Science and Technology of China

{shumings, znie, jrwen}@microsoft.com

fei_c_xing@yahoo.com; mjzhu@ustc.edu

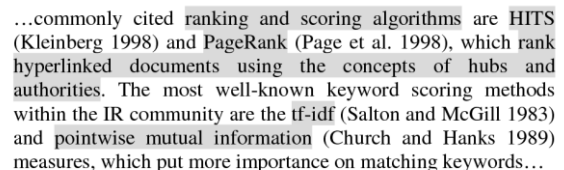
Abstract

Anchor text plays a special important role in improving the performance of general Web search, due to the fact that it is relatively objective description for a Web page by potentially a large number of other Web pages. Academic Search provides indexing and search functionality for academic articles. It may be desirable to utilize anchor text in academic search as well to improve the search results quality. The main challenge here is that no explicit URLs and anchor text is available for academic articles. In this paper we define and automatically assign a *pseudo-URL* for each academic article. And a machine learning approach is adopted to extract *pseudo-anchor* text for academic articles, by exploiting the citation relationship between them. The extracted pseudo-anchor text is then indexed and involved in the relevance score computation of academic articles. Experiments conducted on 0.9 million research papers show that our approach is able to dramatically improve search performance.

1 Introduction

Anchor text is a piece of clickable text that links to a target Web page. In general Web search, anchor text plays an extremely important role in improving the search quality. The main reason for this is that anchor text actually aggregates the opinion (which is more comprehensive, accurate, and objective) of a potentially large number of people for a Web page.

In recent years, academic search (Giles et al., 1998; Lawrence et al., 1999; Nie et al., 2005; Chakrabarti et al., 2006) has become an important supplement to general web search for retrieving research articles. Several academic search systems (including Google Scholar[†], Cite-seer[‡], DBLP[§], Libra^{**}, ArnetMiner^{††}, etc.) have been deployed. In order to improve the results quality of an academic search system, we may consider exploiting the techniques which are demonstrated to be quite useful and critical in general Web search. In this paper, we study the possibility of extracting anchor text for research papers and using them to improve the search performance of an academic search system.



...commonly cited ranking and scoring algorithms are HITS (Kleinberg 1998) and PageRank (Page et al. 1998), which rank hyperlinked documents using the concepts of hubs and authorities. The most well-known keyword scoring methods within the IR community are the tf-idf (Salton and McGill 1983) and pointwise mutual information (Church and Hanks 1989) measures, which put more importance on matching keywords...

Figure 1. An example of one paper citing other papers

The basic search unit in most academic search systems is a research paper. Borrowing the concepts of URL and anchor-text in general Web search, we may need to assign a *pseudo-URL* for one research paper as its identifier and to define the *pseudo-anchor* text for it by the contextual description when this paper is referenced (or mentioned). The pseudo-URL of a research paper could be the combination of its title, authors and publication information. Figure-1 shows an excerpt where one paper cites a couple of other

[†] <http://scholar.google.com/>

[‡] <http://citeseerx.ist.psu.edu/>

[§] <http://www.informatik.uni-trier.de/~ley/db/>

^{**} <http://libra.msra.cn/>

^{††} <http://www.arnetminer.org/>

* This work was performed when Fei Xing and Mingjie Zhu were interns at Microsoft Research Asia.

papers. The grayed text can be treated as the pseudo-anchor text of the papers being referenced. Once the pseudo-anchor text of research papers is acquired, it can be indexed and utilized to help ranking, just as in general web search.

However it remains a challenging task to correctly identify and extract these pseudo-URLs and pseudo-anchor texts. First, unlike the situation in general web search where one unique URL is assigned to each web page as a natural identifier, the information of research papers need to be extracted from web pages or PDF files. As a result, in constructing pseudo-URLs for research papers, we may face the problem of extraction errors, typos, and the case of one research paper having different expressions in different places. Second, in general Web search, anchor text is always explicitly specified by HTML tags (`<a>` and ``). It is however much harder to perform anchor text extraction for research papers. For example, human knowledge may be required in Figure-1 to accurately identify the description of every cited paper.

To address the above challenges, we propose an approach for extracting and utilizing pseudo-anchor text information in academic search to improve the search results quality. Our approach is composed of three phases. In the first phase, each time a paper is cited in another paper, we construct a *tentative* pseudo-URL for the cited paper and extract a *candidate anchor block* for it. The tentative pseudo-URL and the candidate anchor block are allowed to be inaccurate. In the second phase, we merge the tentative pseudo-URLs that should represent the same paper. All candidate anchor blocks belong to the same paper are grouped accordingly in this phase. In the third phase, the final pseudo-anchor text of each paper is generated from all its candidate blocks, by adopting a SVM-based machine learning methodology. We conduct experiments upon a dataset containing 0.9 million research papers. The experimental results show that lots of useful anchor text can be successfully extracted and accumulated using our approach, and the ultimate search performance is dramatically improved when anchor information is indexed and used for paper ranking.

The remaining part of this paper is organized as follows. In Section 2, we describe in detail our approach for pseudo-anchor text extraction and accumulation. Experimental results are reported in Section 3. We discuss related work in Section 4 and finally conclude the paper in Section 5.

2 Our Approach

2.1 Overview

Before describing our approach in detail, we first recall how anchor text is processed in general Web search. Assume that there have been a collection of documents being crawled and stored on local disk. In the first step, each web page is parsed and the out links (or forward links) within the page are extracted. Each link is comprised of a URL and its corresponding anchor text. In the second step, all links are accumulated according to their destination URLs (i.e. the anchor texts of all links pointed to the same URL are merged). Thus, we can get all anchor text corresponding to each web page. Figure-2 (a) demonstrates this process.

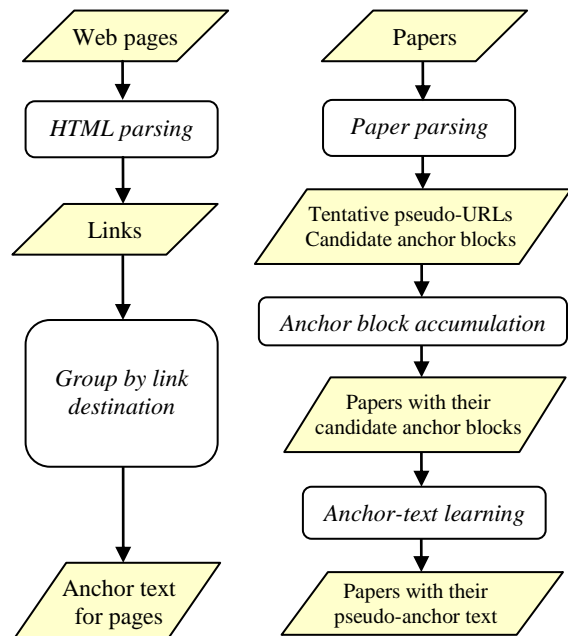


Figure 2. The main process of extracting (a) anchor text in general web search and (b) *pseudo-anchor* text in academic search

For academic search, we need to extract and parse the text content of papers. When a paper *A* mentions another paper *B*, it either explicitly or implicitly displays the key information of *B* to let the users know that it is referencing *B* instead of other papers. Such information can be extracted to construct the *tentative* pseudo-URL of *B*. The pseudo-URLs constructed in this phase are *tentative* because different tentative pseudo-URLs may be merged to generate the same final pseudo-URL. All information related to paper *B* in different papers can be accumulated and treated

as the potential anchor text of *B*. Our goal is to get the anchor text related to each paper.

Our approach for pseudo-anchor text extraction is shown in Figure-2 (b). The key process is similar to that in general Web search for accumulating and utilizing page anchor text. One primary difference between Figure-2 (a) and (b) is the latter accumulates candidate anchor blocks rather than pieces of anchor text. A candidate anchor block is a piece of text that contains the description of one paper. The basic idea is: Instead of extracting the anchor text for a paper directly (a difficult task because of the lack of enough information), we first construct a candidate anchor block to contain the "possible" or "potential" description of the paper. After we accumulate all candidate anchor blocks, we have more information to provide a better estimation about which pieces of texts are anchor texts. Following this idea, our proposed approach adopts a three-phase methodology to extract pseudo-anchor text. In the first phase, each time a paper *B* appearing in another paper *A*, a candidate anchor block is extracted for *B*. All candidate anchor blocks belong to the same paper are grouped in the second phase. In the third phase, the final pseudo-anchor text of each paper is selected among all candidate blocks.

Extracting tentative pseudo-URLs and candidate anchor blocks: When one paper cites another paper, a piece of short text (e.g. "[1]" or "(xxx et al., 2008)") is commonly inserted to represent the paper to be cited, and the detail information (key attributes) of it are typically put at the end of the document (in the references section). We call each paper listed in the references section a *reference item*. The references section can be located by searching for the last occurrence of term 'reference' or 'references' in larger fonts. Then, we adopt a rule-based approach to divide the text in the references section into reference items. Another rule-based approach is used to extract paper attributes (title, authors, year, etc) from a reference item. We observed some errors in our resulting pseudo-URLs caused by the quality of HTML files converted from PDF format, reference item extraction errors, paper attribute extraction errors, and other factors. We also observed different reference item formats for the same paper. The pseudo-URL for a paper is defined according to its title, authors, publisher, and publication year, because these four kinds of information can readily be used to identify a paper.

For each citation of a paper, we treat the sentence containing the reference point (or citation point) as one candidate anchor block. When multiple papers are cited in one sentence, we treat the sentence as the candidate anchor block of every destination paper.

Candidate Anchor Block Accumulation: This phase is in charge of merging all candidate blocks of the same pseudo-URL. As has been discussed, tentative pseudo-URLs are often inaccurate; and different tentative pseudo-URLs may correspond to the same paper. The primary challenge here is perform the task in an *efficient* way and with high *accuracy*. We will address this problem in Subsection 2.2.

Pseudo-Anchor Generation: In the previous phase, all candidate blocks of each paper have been accumulated. This phase is to generate the final anchor text for each paper from all its candidate blocks. Please refer to Subsection 2.3 for details.

2.2 Candidate Anchor Block Accumulation via Multiple Feature-String Hashing

Consider this problem: Given a potentially huge number of tentative pseudo-URLs for papers, we need to identify and merge the tentative pseudo-URLs that represent the same paper. This is like the problems in the record linkage (Fellegi and Sunter, 1969), entity matching, and data integration which have been extensively studied in database, AI, and other areas. In this sub-section, we will first show the major challenges and the previous similar work on this kind of problem. Then a possible approach is described to achieve a trade-off between accuracy and efficiency.

<p>Pseudo-URL 1: Title: Efficient Crawling Through URL Ordering Authors: J Cho, H Garcia-Molina, L Page PubInfo: WWW7 / Computer Networks Year: 1998</p> <p>Pseudo-URL 2: Title: Efficient Crawling Through URL Ordering Authors: J Cho, H Garcia-Molina, L Page PubInfo: In Proceedings of International World Wide Web Conference</p>
--

Figure 3. Two tentative pseudo-URLs representing the same paper

2.2.1 Challenges and candidate techniques

Two issues should be addressed for this problem: similarity measurement, and the efficiency of the algorithm. On one hand, a proper similarity function is needed to identify two tentative pseudo-URLs representing the same paper. Second, the

integration process has to be accomplished efficiently.

We choose to compute the similarity between two papers to be a linear combination of the similarities on the following fields: title, authors, venue (conference/journal name), and year. The similarity function on each field is carefully designed. For paper title, we adopt a term-level edit distance to compute similarity. And for paper authors, person name abbreviation is considered. The similarity function we adopted is fairly well in accuracy (e.g., the similarity between the two pseudo-URLs in Figure-3 is high according to our function); but it is quite time-consuming to compute the similarity for each pair of papers (roughly 10^{12} similarity computation operations are needed for 1 million different tentative pseudo-URLs).

Some existing methods are available for decreasing the times of similarity calculation operations. McCallum et al. (2000) addresses this high dimensional data clustering problem by dividing data into overlapping subsets called canopies according to a cheap, approximate distance measurement. Then the clustering process is performed by measuring the exact distances only between objects from the same canopy. There are also other subspace methods (Parsons et al., 2004) in data clustering areas, where data are divided into subspaces of high dimensional spaces first and then processing is done in these subspaces. Also there are fast blocking approaches for record linkage in Baxter et al. (2003). Though they may have different names, they hold similar ideas of dividing data into subsets to reduce the candidate comparison records. The size of dataset used in the above papers is typically quite small (about thousands of data items). For efficiency issue, Broder et al. (1997) proposed a shingling approach to detect similar Web pages. They noticed that it is infeasible to compare sketches (which are generated by shingling) of all pairs of documents. So they built an inverted index that contains a list of shingle values and the documents they appearing in. With the inverted index, they can effectively generate a list of all the pairs of documents that share any shingles, along with the number of shingles they have in common. They did experiments on a dataset containing 30 million documents.

By adopting the main ideas of the above techniques to our pseudo-URL matching problem, a possible approach can be as follows.

Algorithm Multiple Feature-String Hashing for candidate anchor block accumulation

Input: A list of papers (with their tentative pseudo-URLs and candidate anchor blocks)

Output: Papers with all candidate anchor blocks of the same paper aggregated

```

Initial: An empty hashtable  $h$  (each slot of  $h$  is a list of papers)
For each paper  $A$  in the input list {
  For each feature-string of  $A$  {
    Lookup by the feature-string in  $h$  to get a slot  $s$ ;
    Add  $A$  into  $s$ ;
  }
}
For each slot  $s$  with size smaller than a threshold {
  For any two papers  $A_1, A_2$  in  $s$  {
    float  $fSim = Similarity(A_1, A_2)$ ;
    if( $fSim >$  the specified threshold) {
      Merge  $A_1$  and  $A_2$ ;
    }
  }
}

```

Figure 4. The Multiple Feature-String Hashing algorithm for candidate anchor block accumulation

2.2.2 Method adopted

The method utilized here for candidate anchor block accumulation is shown in Figure 4. The main idea is to construct a certain number of *feature strings* for a tentative pseudo-URL (abbreviated as *TP-URL*) and do hash for the feature strings. A feature string of a paper is a small piece of text which records a part of the paper's key information, satisfying the following conditions: First, multiple feature strings can typically be built from a TP-URL. Second, if two TP-URLs are different representations of the same paper, then the probability that they have at least one common feature string is extremely high. We can choose the term-level n -grams of paper titles (referring to Section 3.4) as feature strings.

The algorithm maintains an in-memory hashtable which contains a lot of slots each of which is a list of TP-URLs belonging to this slot. For each TP-URL, feature strings are generated and hashed by a specified hash function. The TP-URL is then added into some slots according to the hash values of its feature strings. Any two TP-URLs belonging to the same slot are further compared by utilizing our similarity function. If their similarity is larger than a threshold, the two TP-URLs are treated as being the same and therefore their corresponding candidate anchor blocks are merged.

The above algorithm tries to achieve good balance between accuracy and performance. On one hand, compared with the naïve algorithm of performing one-one comparison between all pairs of TP-URLs, the algorithm needs only to compute

the similarity for the TP-URLs that share a common slot. On the other hand, because of the special property of feature strings, most TP-URLs representing the same paper can be detected and merged.

The basic idea of dividing data into overlapped subsets is inherited from McCallum et al. (2000), Broder et al. (1997), and some subspace clustering approaches. Slightly different, we do not count the number of common feature strings between TP-URLs. Common bins (or inverted indices) between data points are calculated in McCallum et al. (2000) as a “cheap distance” for creating canopies. The number of common Shingles between two Web documents is calculated (efficiently via inverted indices), such that Jaccard similarity could be used to measure the similarity between them. In our case, we simply compare any two TP-URLs in the same slot by using our similarity function directly.

The effective and efficiency of this algorithm depend on the selection of feature strings. For a fixed feature string generation method, the performance of this algorithm is affected by the size of each slot, especially the number and size of big slots (slots with size larger than a threshold). Big slots will be discarded in the algorithm to improve performance, just like removing common Shingles in Broder et al. (1997). In Section 4, we conduct experiments to test the performance of the above algorithm with different feature string functions and different slot size thresholds.

2.3 Pseudo-Anchor Text Learning

In this subsection, we address the problem of extracting the final pseudo-anchor text for a paper, given all its candidate anchor blocks (see Figure 5 for an example).

2.3.1 Problem definition

A candidate anchor block is a piece of text with one or some reference points (a reference point is one occurrence of citation in a paper) specified, where a reference point is denoted by a $\langle start_pos, end_pos \rangle$ pair (means start position and end position respectively): $ref = \langle start_pos, end_pos \rangle$. We represent a candidate anchor block to be the following format,

$$AnchorBlock = (Text, ref1, ref2, \dots)$$

We define a *block set* to be a set of candidate anchor blocks for a paper,

$$BlockSet = \{AnchorBlock1, AnchorBlock2, \dots\}$$

Now the problem is: Given a block set containing N elements, extract some text excerpts from them as the anchor text of the paper.

2.3.2 Learn term weights

We adopt a machine-learning approach to assign, for each *term* in the anchor blocks, a discrete *degree* of being anchor text. The main reasons for taking such an approach is twofold: First, we believe that assigning each term a fuzzy degree of being anchor text is more appropriate than a binary judgment as either an anchor-term or non-anchor-term. Second, since the importance of a term for a “link” may be determined by many factors in paper search, a machine-learning could be more flexible and general than the approaches that compute term degrees by a specially designed formula.

Paper	Scaling personalized web search
Candidate-Anchor text	...Haveliwala [8] and Jeh and Widom [9] have done work on efficient personalization, observing that the function mapping reset distributions to stationary distributions is linear...
	...A more recent investigation, [12], uses a different approach: it focuses on user profiles...
	...providing personalized ranking of Web pages based on user preferences, while automating the input generation process for the PPR algorithm [8]...
	...A partial solution to this scaling problem was given in [16], where the dependence from the number of topics...

Figure 5. The candidate pseudo-anchor blocks of a paper

The features used for learning are listed in Table-1.

We observed that it would be more effective if some of the above features are normalized before being used for learning. For a term in candidate anchor block B , its TF are normalized by the BM25 formula (Robertson et al., 1999),

$$TF_{norm} = \frac{(k_1 + 1) \cdot TF}{k_1 \cdot (b + (1 - b) \cdot \frac{|B|}{L}) + TF}$$

where L is average length of the candidate blocks, $|B|$ is the length of B , and k_1, b are parameters.

DF is normalized by the following formula,

$$IDF = \log(1 + \frac{N}{DF})$$

where N is the number of elements in the block set (i.e. total number of candidate anchor blocks for the current paper).

Features $RefPos$ and $Dist$ are normalized as,

$$RefPos_{norm} = RefPos / |B|$$

$$Dist_{norm} = (Dist - RefPos) / |B|$$

And the feature $BlockLen$ is normalized as,

$$BlockLen_{norm} = \log(1+BlockLen)$$

Features	Description
<i>DF</i>	Document frequency: Number of candidate blocks in which the term appears, counted among all candidate blocks of all papers. It is used to indicate whether the term is a stop word or not.
<i>BF</i>	Block frequency: Number of candidate blocks in which the term appears, counted among all candidate blocks of this paper.
<i>CTF</i>	Collection term frequency: Total number of times the term appearing in the blocks. For multiple times of occurrences in one block, all of them are counted.
<i>IsInURL</i>	Specify whether the term appears in the pseudo-URL of the paper.
<i>TF</i>	Term frequency: Number of times the terms appearing in the candidate block.
<i>Dist</i>	Directed distance from the nearest reference point to the term location
<i>RefPos</i>	Position of the nearest reference point in the candidate pseudo-anchor block.
<i>BlockLen</i>	Length of the candidate pseudo-anchor block

Table 1. Features for learning

We set four term importance levels, from 1 (unrelated terms or stop words) to 4 (words participating in describing the main ideas of the paper).

We choose support vector machine (SVM) for learning term weights here, because of its powerful classification ability and well generalization ability (Burges, 1998). We believe some other machine learning techniques should also work here. The input of the classifier is a feature vector of a term and the output is the importance level of the term. Given a set of training data $\{feature_i, level_i\}_{i=1}^l$, a decision function $f(x)$ can be acquired after training. Using the decision function, we can assign an importance level for each term automatically.

3 Experiments

3.1 Experimental Setup

Our experimental dataset contains 0.9 million papers crawled from the web. All the papers are processed according to the process in Figure-2 (b). We randomly select 300 queries from the query log of Libra (libra.msra.cn) and retrieve the results in our indexing and ranking system with/without the pseudo-anchors generated by our approach. Then the volunteer researchers and students in our group are involved to judge the search results. The top 30 results of different ranking algorithms for each query are labeled and assigned a relevance value from 1 (meaning 'poor match') to 5 (meaning 'perfect match'). The

search results quality is measured by NDCG (Jarvelin and Kekalainen, 2000).

3.2 Overall Effect of our Approach

Figure 6 shows the performance comparison between the results of two baseline paper ranking algorithms and the results of including pseudo-anchor text in ranking.

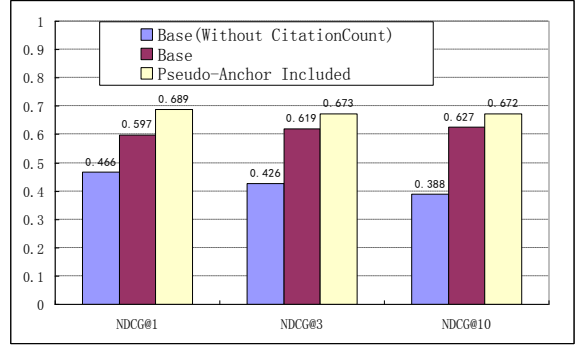


Figure 6. Comparison between the baseline approach and our approach (measure: nDCG)

The “Base” algorithm considers the title, abstract, full-text and static-rank (which is a function of the citation count) of a paper. In a bit more detail, for each paper, we adopt the BM25 formula (Robertson et al., 1999) over its title, abstract, and full-text respectively. And then the resulting score is linearly combined with the static-rank to get its final score. The static-rank is computed as follows,

$$StaticRank = \log(1+CitationCount) \quad (3.1)$$

To test the performance of including pseudo-anchor text in ranking, we compute an anchor score for each paper and linearly combine it with its baseline score (i.e. the score computed by the baseline algorithm).

We tried two kinds of ways for anchor score computation. The first is to merge all pieces of anchor excerpts (extracted in the previous section) into a larger piece of anchor text, and use BM25 to compute its relevance score. In another approach called homogeneous evidence combination (Shi et al., 2006), a relevance score is computed for each anchor excerpt (still using BM25), and all the scores for the excerpts are sorted descending and then combined by the following formula,

$$S_{anchor} = \sum_{i=1}^m \frac{1}{(1+c \cdot (i-1))^2} \cdot s_i \quad (3.2)$$

where s_i ($i=1, \dots, m$) are scores for the m anchor excerpts, and c is a parameter. The primary idea

here is to let larger scores to have relative greater weights. Please refer to Shi et al. (2006) for a justification of this approach. As we get slightly better results with the latter way, we use it as our final choice for computing anchor scores.

From Figure 6, we can see that the overall performance is greatly improved by including pseudo-anchor information. Table 2 shows the t-test results, where a “>” indicates that the algorithm in the row outperforms that in the column with a p -value of 0.05 or less, and a “>>” means a p -value of 0.01 or less.

	Base	Base (without CitationCount)
Our approach	>	>>
Base		>>
Base (without CitationCount)		

Table 2. Statistical significance tests (t-test over nDCG@3)

Table 3 shows the performance comparison by using some traditional IR measures based on binary judgments. Since the results of not including CitationCount are much worse than the other two, we omit it in the table.

Measure	MAP	MRR	P@1	P@10
Base (including CitationCount)	0.364	0.727	0.613	0.501
Our Approach	0.381	0.734	0.625	0.531

Table 3. Performance comparison using binary judgment measures

3.3 Sample Query Analysis

Here we analyze some sample queries to get some insights about why and how pseudo-anchor improves search performance. Figure-7 and Figure-8 show the top-3 results of two sample queries: {TF-IDF} and {Page Rank}.

For query "TF-IDF", the top results of the baseline approach have keyword "TF-IDF" appeared in the title as well as in other places of the papers. Although the returned papers are relevant to the query, they are not excellent because typically users may want to get the first TF-IDF paper or some papers introducing TF-IDF. When pseudo-anchor information is involved, some excellent results (B1, B2, B3) are generated. The main reason for getting the improved results is that these papers (or books) are described with "TF-IDF" when lots of other papers cite them.

A1. K Sugiyama, K Hatano, M Yoshikawa, S Uemura. Refinement of TF-IDF schemes for web pages using their hyperlinked neighboring pages. Hypertext'03
A2. A Aizawa. An information-theoretic perspective of tf-idf measures. IPM'03.
A3. N Oren. Reexamining tf.idf based information retrieval with Genetic Programming. SAICSIT'02.
(a) Without anchor
B1. G Salton, MJ McGill. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.
B2. G Salton and C Buckley. Term weighting approaches in automatic text retrieval. IPM'98.
B3. R Baeza-Yates, B Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley, 1999
(b) With anchor

Figure 7. Top-3 results for query TF-IDF

A1. V Safronov, M Parashar, Y Wang et al. Optimizing Web servers using Page rank prefetching for clustered accesses. Information Sciences. 2003.
A2. AO Mendelzon, D Rafiei. An autonomous page ranking method for metasearch engines. WWW, 2002.
A3. FB Kalhoff. On formally real Division Algebras and Quasifields of Rank two.
(a) Without anchor
B1. S Brin, L Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. WWW, 1998
B2. L Page, S Brin, R Motwani, T Winograd. The pagerank citation ranking: Bringing order to the web. 1998.
B3. JM Kleinberg. Authoritative sources in a hyperlinked environment. Journal of the ACM, 1999.
(b) With anchor

Figure 8. Top-3 results for query Page Rank

Figure-8 shows another example about how pseudo-anchor helps to improve search results quality. For query "Page Rank" (note that there is a space in between), the results returned by the baseline approach are not satisfactory. In the papers returned by our approach, at least B1 and B2 are very good results. Although they did not label themselves "Page Rank", other papers do so in citing them. Interestingly, although the result B3 is not about the "PageRank" algorithm, it describes another popular "Page Rank" algorithm in addition to PageRank.

Another interesting observation from the two figures is that our approach retrieves older papers than the baseline method, because old papers tend to have more anchor text (due to more citations). So our approach may not be suitable for retrieve newer papers. To overcome this problem, maybe publication year should be considered in our ranking functions.

3.4 Anchor Accumulation Experiments

We conduct experiments to test the effectiveness and efficiency of the multiple-feature-string-hashing algorithm presented in Section 2.2. The duplication detection quality of this algorithm is determined by the appropriate selection of fea-

ture strings. When feature strings are fixed, the slot size threshold can be used to tune the tradeoff between accuracy and performance.

Feature Strings Slot Distr.	Ungram	Bigram	Trigram	4-gram
# of Slots	$1.4*10^5$	$1.2*10^6$	$2.8*10^6$	$3.4*10^6$
# of Slots with size > 100	5240	6806	1541	253
# of Slots with size > 1000	998	363	50	5
# of Slots with size > 10000	59	11	0	0

Table 4. Slot distribution with different feature strings

We take all the papers extracted from PDF files as input to run the algorithm. Identical TP-URLs are first eliminated (therefore their candidate anchor blocks are merged) by utilizing a hash table. This pre-process step results in about 1.46 million distinct TP-URLs. The number is larger than our collection size (0.9 million), because some cited papers are not in our paper collection. We tested four kinds of feature strings all of which are generated from paper title: unigrams, bigrams, trigrams, and 4-grams. Table-4 shows the slot size distribution corresponding to each kind of feature strings. The performance comparison among different feature strings and slot size thresholds is shown in Table 5. It seems that bigrams achieve a good trade-off between accuracy and performance.

Feature Strings	Slot Size Threshold	Dup. papers Detected	Processing Time (sec)
Unigram	5000	529,717	119,739.0
	500	327,357	7,552.7
Bigram	500	528,981	8,229.6
Trigram	Infinite	518,564	8,420.4
	500	516,369	2,654.9
4-gram	500	482,299	1,138.2

Table 5. Performance comparison between different feature strings and slot size thresholds

4 Related Work

There has been some work which uses anchor text or their surrounding text for various Web information retrieval tasks. It was known at the very beginning era of internet that anchor text was useful to Web search (McBryan, 1994). Most Web search engines now use anchor text as primary and power evidence for improving search performance. The idea of using contextual text in a certain vicinity of the anchor text was proposed in Chakrabarti et al. (1998) to automatically compile some lists of authoritative Web

resources on a range of topics. An anchor window approach is proposed in Chakrabarti et al (1998) to extract implicit anchor text. Following this work, anchor windows were considered in some other tasks (Amitay et al., 1998; Haveliwala et al., 2002; Davison, 2002; Attardi et al., 1999). Although we are inspired by these ideas, our work is different because research papers have many different properties from Web pages. From the viewpoint of implicit anchor extraction techniques, our approach is different from the anchor window approach. The anchor window approach is somewhat simpler and easy to implement than ours. However, our method is more general and flexible. In our approach, the anchor text is not necessarily to be in a window.

Citeseer (Giles et al., 1998; Lawrence et al., 1999) has been doing a lot of valuable work on citation recognition, reference matching, and paper indexing. It has been displaying contextual information for cited papers. This feature has been shown to be helpful and useful for researchers. Differently, we are using context description for improving ranking rather than display purpose. In addition to Citeseer, some other work (McCallum et al., 1999; Nanba and Okumura, 1999; Nanba et al., 2004; Shi et al., 2006) is also available for extracting and accumulating reference information for research papers.

5 Conclusions and Future Work

In this paper, we propose to improve academic search by utilizing pseudo-anchor information. As pseudo-URL and pseudo-anchor text are not as explicit as in general web search, more efforts are needed for pseudo-anchor extraction. Our machine-learning approach has proven successful in automatically extracting implicit anchor text. By using the pseudo-anchors in our academic search system, we see a significant performance improvement over the basic approach.

Acknowledgments

We would like to thank Yunxiao Ma and Pu Wang for converting paper full-text from PDF to HTML format. Jian Shen has been helping us do some reference extraction and matching work. Special thanks are given to the researchers and students taking part in data labeling.

References

- E. Amitay. 1998. Using common hypertext links to identify the best phrasal description of target web documents. In Proc. of the SIGIR'98 Post Conference Workshop on Hypertext Information Retrieval for the Web, Melbourne, Australia.
- G. Attardi, A. Gulli, and F. Sebastiani. 1999. Theseus: categorization by context. In Proceedings of the 8th International World Wide Web Conference.
- A. Baxter, P. Christen, T. Churches. 2003. A comparison of fast blocking methods for record linkage. In ACM SIGKDD'03 Workshop on Data Cleaning, Record Linkage and Object consolidation. Washington DC.
- A. Broder, S. Glassman, M. Manasse, and G. Zweig. 1997. Syntactic clustering of the Web. In Proceedings of the Sixth International World Wide Web Conference, pp. 391-404.
- C.J.C. Burges. 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2, 121-167.
- S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. 1998. Automatic resource list compilation by analyzing hyperlink structure and associated text. In Proceedings of the 7th International World Wide Web Conference.
- K. Chakrabarti, V. Ganti, J. Han, and D. Xin. 2006. Ranking objects based on relationships. In SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data, pages 371–382, New York, NY, USA. ACM.
- B. Davison. 2000. Topical locality in the web. In SIGIR'00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, pages 272-279, New York, NY, USA. ACM.
- I.P. Fellegi, and A.B. Sunter. A Theory for Record Linkage, *Journal of the American Statistical Association*, 64, (1969), 1183-1210.
- C. L. Giles, K. Bollacker, and S. Lawrence. 1998. CiteSeer: An automatic citation indexing system. In Ian Witten, Rob Akscyn, and Frank M. Shipman III, editors, *Digital Libraries 98 - The Third ACM Conference on Digital Libraries*, pages 89–98, Pittsburgh, PA, June 23–26. ACM Press.
- T.H. Haveliwala, A. Gionis, D. Klein, and P. Indyk. 2002. Evaluating strategies for similarity search on the web. In WWW '02: Proceedings of the 11th international conference on World Wide Web, pages 432–442, New York, NY, USA. ACM.
- K. Jarvelin, and J. Kekalainen. 2000. IR Evaluation Methods for Retrieving Highly Relevant Documents. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR2000).
- S. Lawrence, C.L. Giles, and K. Bollacker. 1999. Digital libraries and Autonomous Citation Indexing. *IEEE Computer*, 32(6):67–71.
- A. McCallum, K. Nigam, J. Rennie, and K. Seymore. 1999. Building Domain-specific Search Engines with Machine Learning Techniques. In Proceedings of the AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace.
- A. McCallum, K. Nigam, and L. Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In Proc. 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining.
- O.A. McBryan. 1994. Genvl and www: Tools for taming the web. In Proceedings of the First International World Wide Web Conference, pages 79-90.
- H. Nanba, M. Okumura. 1999. Towards Multi-paper Summarization Using Reference Information. In Proc. of the 16th International Joint Conference on Artificial Intelligence, pp.926-931.
- H. Nanba, T. Abekawa, M. Okumura, and S. Saito. 2004. Bilingual PRESRI: Integration of Multiple Research Paper Databases. In Proc. of RIAO 2004, 195-211.
- L. Parsons, E. Haque, H. Liu. 2004. Subspace clustering for high dimensional data: a review. *SIGKDD Explorations* 6(1): 90-105.
- S.E. Robertson, S. Walker, and M. Beaulieu. 1999. Okapi at TREC-7: automatic ad hoc, filtering, VLC and filtering tracks. In Proceedings of TREC'99.
- S. Shi, R. Song, and J-R Wen. 2006. Latent Additivity: Combining Homogeneous Evidence. Technique report, MSR-TR-2006-110, Microsoft Research, August 2006.
- S. Shi, F. Xing, M. Zhu, Z. Nie, and J.-R. Wen. 2006. Pseudo-Anchor Extraction for Search Vertical Objects. In Proc. of the 2006 ACM 15th Conference on Information and Knowledge Management. Arlington, USA.
- Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma. 2005. Object-level ranking: bringing order to web objects. In WWW'05: Proceedings of the 14th international conference on World Wide Web, pages 567–574, New York, NY, USA. ACM.