Pseudo-Anchor Text Extraction for Vertical Search

Shuming Shi, Fei Xing[1], Mingjie Zhu[2], Zaiqing Nie, Ji-Rong Wen
Microsoft Research Asia, Beijing, China
[1]Department of Computer Science, Beijing Normal University, Beijing, China
[2]Department of Computer Science, University of Science and Technology of China
{shumings, znie, jrwen}@microsoft.com, [1]fxing@cs.bnu.edu.cn, [2]mjzhu@ustc.edu

ABSTRACT
Anchor text plays a special important role in improving the performance of general Web search. The importance of anchor text comes from the fact that it is fairly objective description for a Web page by potentially a large amount of other Web pages. Vertical search provides indexing and search functionality for objects in a certain domain, and is becoming an important supplement for general Web search. It is desired to utilize anchor text in vertical search as well to improve search performance. Vertical objects typically lack explicit URLs to accurately identify them. The anchor-text of a vertical object is also hard to acquire explicitly. This paper proposes concepts of *pseudo-URL* and *pseudo-anchor-text* for vertical objects, corresponding to the URL and anchor-text of a general Web page. For extracting and utilizing pseudo-anchor-text information of vertical objects, we focus on candidate anchor block accumulation and pseudo-anchor extraction in this paper. State-of-the-art data integration techniques are utilized to accumulate candidate anchor blocks belonging to same objects. Pseudo-anchor text for each object is extracted from its candidate anchor blocks using a machine learning based approach. A case study in academic search domain indicates that our approach is able to dramatically improve search performance.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Search process; H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms
Algorithms, Experimentation

Keywords
Anchor extraction, Vertical search, Vertical objects, Pseudo-URL, Pseudo-Anchor text, Implicit anchor-text

1. INTRODUCTION
Modern search engines commonly provide two categories of search functionalities to users: general Web search, and vertical search. With vertical search, we mean searching for a class of Web objects or information in a certain domain. Examples of vertical search include product search, image search, academic search, book search, people search, and others. Vertical search has become an important supplement for general Web search. Different from general Web search, vertical search commonly deals with information about certain types of real-world *objects* instead of general Web pages. One document may be describing multiple objects, and the same object can appear in multiple documents of different types (e.g. HTML Web pages, XML files, PDF files, emails, and instant messages). For example, Figure 1 shows a Web page containing information of a list of books. And in Figure 2, information about the product "Dell Latitude C640" appears in various Web sites. Lots of current vertical search engines extract object information from the Web and provide indexing and search services for objects. For example, all the structured product data of the Windows Live Product Search (products.live.com) and a portion of the data in Froogle (froogle.google.com) are extracted from the Web, and ZoomInfo (www.zoominfo.com) extracts people information from multiple Web pages and integrates them. As vertical search typically focuses on a specific domain or specific types of objects, it enjoys greater odds of providing rich, precise, and structured information to users by utilizing domain knowledge.

Figure 1. A Web page containing information of some books (from http://www.amazon.co.uk)

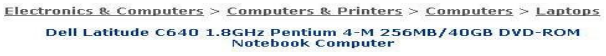
State-of-the-art general Web search techniques may be adequate for finding desired Web pages. However, for most vertical search domains, their search performances still have substantial room for improvement. In addition to the difficulty of extracting structured information from the Web, one reason for this is that some techniques, which are demonstrated to be quite useful and critical in general Web search, have not yet been applied to most vertical object search engines. 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. As a result, such information is less susceptible to spam[1].

[1] Link bombing, of course, aims at page ranking and anchor text. However, they are still much harder to be affected than page content. Most Web page designers could alter their page

To improve vertical search performance, anchor text would be highly useful if applied to vertical search as well. Borrowing the concepts of URL and anchor-text in general Web search, we may need to assign a *pseudo-URL* for a vertical object as its identifier and to define the *pseudo-anchor text* for it by the contextual description around its pseudo-URL when this object is referenced (mentioned). For example, "The Fall of the Roman Empire ~ Peter Heather" can be thought of as the pseudo-URL of the first book in Figure 1. In Figure 2, most of the displayed text can be treated as anchor text of object "Dell Latitude C640." Once the pseudo-URL and pseudo-anchor text of each object are determined, anchor text techniques can be applied in ranking vertical objects. However, pseudo-URL and pseudo-anchor are much different from their counterparts.

(a) An excerpt from a page in http://froogle.google.com

(b) A page excerpt (from http://pcworld.about.com/news/Jan272003id108954.htm)



(c) An excerpt from a page in http://www.overstock.com

Figure 2. Product "Dell Latitude C640" appears in various Web pages

First, the URL of a Web page is assigned by human beings to act as a natural identifier of the page. However, for vertical objects automatically extracted from Web pages or other kinds of documents, we need to construct *pseudo-URLs* according to the information extracted. Some pseudo-URLs may not be accurate because of extraction errors. In addition, because an object can have different descriptions on varying Web pages, two different pseudo-URLs may correspond to the same object. For example, "Dell C640" and "Dell Latitude C640" actually represent the same object.

Second, in general Web search, anchor text is always *explicitly* specified by Web page designers via HTML tags (<a> and). This kind of *explicit* anchor text can easily be extracted and used for Web search. However, it is not the case for some vertical search domains. The pseudo-anchor text of a vertical object is often implicit and it needs to make efforts to extract them. For example, although most pieces of text in Figure 2 are related to Dell C640, their relationship is not explicitly specified.

Due to the above differences, it is a challenge to collect the pseudo-anchor text corresponding to a vertical object.

contents freely, while much more efforts should be made to modify other people's opinions.

This paper studies the problem of the extraction and aggregation of implicit anchor text for vertical objects. We propose an approach for extracting and utilizing pseudo-anchor-text information of vertical objects to improve vertical object search. This approach adopts a three-phase methodology to extract pseudo-anchors. In the first phase, each time an object appearing in a document, its pseudo-URL is identified and a candidate anchor block is extracted for the object. All candidate anchor blocks belong to the same object are grouped in the second phase. In the third phase, the ultimate pseudo-anchor text of each object is extracted from the aggregated candidate blocks. The last two phases are focused in this paper. State-of-the-art data integration techniques are utilized in the second phase to accumulate candidate anchor blocks belonging to same objects. And a machine-learning method is proposed to automatically assign each term in each candidate block a degree of belonging to anchor text, given all information provided by the candidate blocks.

We take the process of extracting pseudo-anchor text for research paper objects as an example to illustrate how to apply our proposed approach to a specific domain. We did some experiments based on our paper search system called Libra (please refer to [23] for a description of the system) and evaluated the performance by using queries selected from its query log. Experimental results show that lots of useful anchor text information can be successfully extracted and accumulated using our approach, and ultimate search performance can be dramatically improved when anchor information is used.

The remaining part of this paper is organized as follows. In Section 2, we describe in detail our approach for pseudo-anchor text extraction, accumulation, and utilization. A case study in the academic domain for search research publications is illustrated in Section 3. In Section 4, we show the experimental results in the academic domain based on our research prototype Libra. Related work is discussed in Section 5. We then provide our conclusion and discuss future work.

2. OUR APPROACH

As we have illustrated in the previous section, because of the special properties of pseudo-URL and pseudo-anchor text in vertical object search, some work become more complicated than processing the anchor text of Web pages. In this section, we propose our approach for pseudo-anchor extraction, gathering and utilization in vertical search. Some key components in the approach will be discussed in this section as well.

2.1 Overview

Before describing this approach in detail, we first recall how anchor text is processed in general Web search (Figure-3(a)). Assume that there have been a collection of documents being crawled and placed into local disk. In the first step, each Web page is parsed (via an HTML parser) and out-links within it are extracted. Each link is comprised of a URL and its corresponding anchor text. Then, in the second step, all links are accumulated according to destination URLs (i.e. the anchor text of all links with the same destination URL is merged). Thus, we can get all anchor text corresponding to each Web page. Figure 3(a) shows the process.

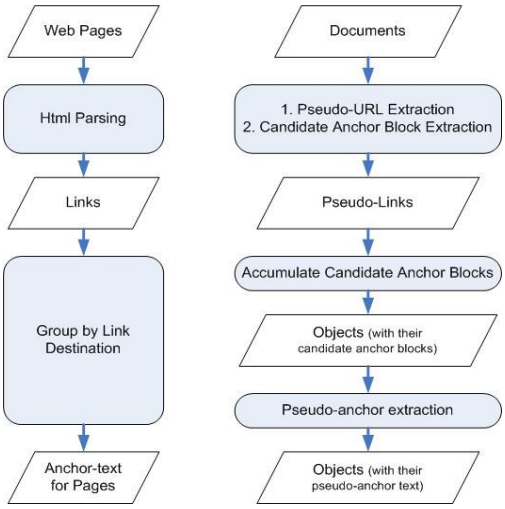


Figure 3. Anchor processing in general Web search and vertical object search (a) Anchor processing in general Web search; (b) Our framework for pseudo-anchor extraction.

For vertical object search, we need to extract object information from documents. When a document *D* mentions an object *A*, it will explicitly or implicitly display the key information of *A* to let users know that it is discussing *A* instead of other objects. Ideally, this key information can be extracted to construct the pseudo-URL of the object. Pseudo-URL will be used for merging all kinds of information belonging to one object. In addition to the key information, additional descriptions of the object may exist as well. All information (key or ordinary) related to object *A* in document *D* can be treated as an *anchor item* of object *A*. Our goal in this paper is to get all anchor items related to a vertical object.

Our main approach for pseudo-anchor text extraction is shown in Figure 3 (b). The approach is similar to that in general Web search for accumulating and utilizing page anchor text. One primary difference between Figure 3(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 possible descriptions of an object. The basic idea is: Instead of extracting the anchor text for an object directly (a difficult task because of lack of enough information), we first construct a candidate anchor block to contain "possible" descriptions. After we accumulate all candidate anchor blocks, we have more information to provide a better estimation about which pieces of text is anchor text. Following this idea, our proposed approach adopts a three-phase methodology to extract pseudo-anchors. In the first phase, for each time an object appearing in a document, a candidate anchor block is extracted for the object. All candidate anchor blocks belong to the same object are grouped in the second phase. In the third phase, the ultimate pseudo-anchor text of each object is extracted by considering all candidate blocks of the object.

This approach contains four modules: pseudo-URL extraction, candidate anchor block extraction, candidate anchor block accumulation, and pseudo-anchor extraction. For most vertical object types, some of these steps may be challenging. Fortunately, there are some existing techniques that can aid in implementation of some of these modules. In the following subsections, we will briefly analyze main functionalities, implementation challenges, and existing techniques for each module.

2.1.1 Pseudo-URL extraction

This module is for identifying and extracting vertical objects from a document. This step is actually doing entity extraction [18][19] or information extraction (especially Web information extraction) [20][21][22]. Many techniques have been proposed to address this problem, from rule-based methods to machine-learning approaches. This step may be quite simple for some kinds of vertical objects (e.g. Web images, research papers in well-formatted research documents, etc). However, for vertical objects such as products and people, this step becomes a greater challenge. We have built some Web object extraction technologies to efficiently identify the key attributes of vertical objects (i.e. their pseudo-URLs) [27][28]. Object information extraction is not our focus in this paper.

2.1.2 Candidate anchor block extraction

This module extracts some pieces of text as candidate anchor block of an object. If we call one occurrence of an object in a document a *reference point*, then the surrounding text around a reference point is commonly related to the object referenced. Therefore one primary way to construct a candidate block is surrounding text extraction. The simplest surrounding text extraction approach is to consider all text with distance smaller than a threshold. Image search engines extract surrounding text for images by analyze HTM tags. As an extreme case, the entire document (or paragraph) containing an object can be considered as a candidate anchor block of the object[2]. For HTML documents, our vision-based Web page segmentation technology could be utilized to extract the anchor blocks [29].

2.1.3 Candidate anchor block accumulation

After all documents are processed, all the extracted candidate anchor blocks will be accumulated according to their related pseudo-URLs. This module is in charge of merging all candidate blocks of the same vertical object.

As has been discussed, pseudo-URLs are often inaccurate descriptions of vertical objects. Different pseudo-URLs may correspond to the same object. The challenge in this module is how to merge different pseudo-URLs of the same object in an effective way and with high accuracy. We will address this problem in Subsection 2.2.

2.1.4 Pseudo-anchor extraction

In the previous module, all candidate blocks of each object have been accumulated. This module is for extract anchor text for each object based on all its candidate blocks. Our main approach for implementing this module will be described in Subsection 2.3.

The last two modules are our focus in this paper. We will first demonstrate how to utilize state-of-the-art data integration techniques for merging candidate anchor blocks belonging to one object. Then a machine learning based anchor text extraction method will be illustrated.

---

[2] This does not mean they are valid anchor text. The ultimate anchor text for an object will be extracted in the last module.

2.2 Candidate Anchor Block Accumulation

Consider this problem: given a huge number of pseudo-URLs for vertical objects, we have to identify and merge pseudo-URLs that represent the same object. This is like problems in the record linkage [11], 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

Pseudo-URL 1:
Title: E#icient Crawling Through URL Ordering
Authors: J Cho, H Garcia-Molina, L Page
PubInfo: WWW7 / Computer Networks
Year: 1998
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

Figure 4. Two pseudo-URLs representing the same paper object

Pseudo-URL 1: Dell Latitude C640
Pseudo-URL 2: Dell C640
Pseudo-URL 3: Dell Latitude C640 Laptop

Figure 5. Three pseudo-URLs representing the same product object

2.2.1 Challenges and candidate techniques

Two issues are addressed for this problem: similarity measurement, and the efficiency of the algorithm. First, a proper similarity function is required to identify two pseudo-URLs represent the same object. Second, the integration process has to be accomplished efficiently.

Given a vertical domain, a domain-specific similarity function can be defined to calculate the similarity between any pair of pseudo-URLs. When the similarity value is larger than a threshold, then there is a high probability that the two objects are the same. For example, a carefully designed similarity function is capable of determining that the two paper objects in Figure 4 are actually the same paper. In Figure 5, we can determine that the three pseudo-URLs represent the same object, if domain knowledge is adopted. Edit distance [17] is a commonly used domain-independent similarity function. There is some work [12] proposing adaptive entity matching functions that can be trained to obtain better performance in a particular domain.

In our scenarios, large numbers of pseudo-URLs are required for processing, so the efficiency is especially important. It could take too much time to compute the similarity of every pair of pseudo-URLs (roughly 10^{12} similarity computation operations are needed for 1 million objects). There are some existing methods for decreasing similarity calculation operations. McCallum, Nigam and Unger [13] try to solve 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. And inverted index can be used as a cheap distance metric to construct canopies. There are also many other subspace methods [14] in data clustering areas. 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 [15]. 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 [16] 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 combining some of the above techniques and applying them to the pseudo-URL matching problem, a possible approach can be as follows.

Algorithm Multiple Feature-String Hashing for candidate block accumulation

Input: A list of objects (with their pseudo-URLs and candidate anchor blocks)
Output: A list of objects, with all candidate anchor blocks of the same object aggregated

```
Initial: An empty hashtable h (each slot of h is a list of objects)
For each object 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 objects A1, A2 in s {
      float fSim = Similarity(A1, A2);
      if(fSim > the specified threshold) {
         Merge A1 and A2;
      }
   }
}
```

Figure 6. The Multiple Feature-String Hashing algorithm for candidate anchor block accumulation

2.2.2 Method adopted

The method utilized in our candidate block accumulation module is shown in Figure 6. The main idea is to construct a certain number of *feature strings* for a pseudo-URL and do hash for the feature strings. A feature string of an object is a small piece of text which records a part of the object's key information. A pseudo-URL typically has several feature strings within. If two pseudo-URLs are essentially different representations of the same object, then the probability that they have at least one common feature string is extremely high[3]. Different kinds of vertical objects may need to define different kinds of feature strings. For example, for book objects, feature strings can be n-grams of book titles.

[3] Please note that two objects are not necessarily similar if they have one feature string in common.

The algorithm maintains an in-memory hash-table which contains a lot of slots each of which is a list of pseudo-URLs belonging to this slot. For each pseudo-URL, feature strings are generated and hashed by a specified hash function. The pseudo-URL is then added into some slots according to the hash values of its feature strings. Any two pseudo-URLs belonging to the same slot are further compared by utilizing a carefully designed similarity function. If their similarity is larger than a threshold, then the two pseudo-URLs are thought of as being the same and therefore their 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 pseudo-URLs, the algorithm needs only to compare pseudo-URLs that share a common slot. On the other hand, because of the special property of feature strings, most pseudo-URLs representing the same object can be detected and merged.

The basic idea of dividing data into overlapped subsets is inherited from [13], [16], and some subspace clustering approaches. Slightly different from [13] and [16], we do not count the number of common feature strings between pseudo-URLs. Common bins (or inverted indices) between data points are calculated in [13] 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 do not need this. We simply compare any two pseudo-URLs in the same slot by using domain-specific similarity functions directly.

The duplication detection quality of this algorithm is determined by the appropriate selection of feature strings. And the performance of this algorithm depends on size of each slot, especially the number and size of big slots. Big slots (slots with size larger than a threshold) will be discarded in the algorithm to improve the performance, just like removing common Shingles in [16]. In Section 4, we will use experiments to test the performance of the above algorithm with different feature string functions and different slot size thresholds.

2.3 Pseudo-Anchor Text Extraction by Machine Learning

Before entering into this step, all candidate anchor blocks for each object have been accumulated. In this subsection, we address the problem of extracting pseudo-anchor text for an object, given all its candidate anchor blocks.

For objects of different types and in different domains, their candidate anchor blocks may share some common structures. Most candidate anchor blocks may be pieces of text with **reference points** (a reference point is one occurrence of an object in a document) within. Table 1 and 2 list a paper object (in academic search domain) and a product object (in the product search domain). Reference points are labeled by bold and italic text.

Since objects in a lot of different domains can have the above kind of anchor block representation, we make the following problem definition.

2.3.1 Problem definition

We define a candidate anchor block as a piece of text with one or some reference points specified, where a reference point is denoted by a <*start_pos*, *end_pos*> pair (means start position and end position respectively): *ref* = <*start_pos*, *end_pos*>. That is, we represent a candidate anchor block as the following format,

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

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

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

The problem is: Given a block set containing *N* elements, extract some text excerpts from them to satisfy some desired conditions.

Here a block set is used to model all candidate anchor blocks of an object. And our goal is to extract text excerpts best describing the object. Please note that it may be possible that more than one text excerpts are extracted from one anchor block.

Table 1. A research paper object and its candidate anchor text

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…
	… …

Table 2. A product object and its candidate anchor text

Product	Dell Latitude C640
Candidate-Anchor text	…***Dell C640*** Intel Pentium 4-M 2GHz / 14.1-Inch / 256MB DDR / 20HDD / CD-ROM / Windows XP Pro... Price: $649.99 Description from SHOP.COM: TigerDirect.com: ***Dell Latitude C640*** Refurbished Notebook PC Dell Latitude notebooks meet the wide-ranging needs of business and institutional customers - needs that include powerful performance, portability, and flexibility…
	…***Dell Latitude C640*** WHAT'S HOT: The ***Latitude C640***--a refresh of another Dell business-focused laptop, the Latitude C610--has a modular bay located conveniently on the front that can hold an optical drive…
	…Electronics & Computers > Computers & Printers > Computers > Laptops ***Dell Latitude C640*** 1.8GHz Pentium 4-M 256MB/40GB DVD-ROM Notebook Computer…
	… …

2.3.2 Learn term weights

Given the above problem, multiple ways may be available for solving it. Instead of determining whether a piece of text is anchor text, we adopt a machine-learning approach to assign a discrete *degree* for each term as to its anchor properties in each candidate anchor block. The main reasons for taking such an approach is twofold: First, we believe that assigning each term a fuzzy degree as a potential anchor 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 some vertical search domains, a machine-learning approach is required to combine all of them. A machine-learning approach can be more flexible and general than approaches that compute term degrees by a specially designed formula.

Thus, for each term in every candidate anchor block, our goal is to learn a degree of belonging to anchor text for it, given all information provided by the candidate blocks.

To adopt a machine-learning approach, a classifier should be selected and some features should be generated for one term. Some training data should also be generated for user labeling.

2.3.2.1 Features for learning

To apply a machine-learning algorithm, some features should be extracted for each term in a candidate block. Possible features are shown in Table 3.

Table 3. Global features extracted

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

We find that it would be more effective if some of the above features are normalized before they are used for learning. For a term in candidate anchor block B, its TF are normalized by BM25 formula [1],

$$TF_{norm} = \frac{(k_1 + 1) \cdot tf}{k_1 \cdot (b + (1-b) \cdot \frac{|B|}{L}) + tf} \qquad (2.1)$$

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 Number of elements in the block set (i.e. total number of candidate anchor blocks for current object).

Feature *RefPos* and *Dist* are normalized respectively as follows,

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

And feature *BlockLen* is normalized as,

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

2.3.2.2 Labeling

We set four term importance levels, from 1 (unrelated terms or stop words) to 4 (words those participate in describing the main properties of the object). Users cannot label terms with too many levels.

2.3.2.3 Learning algorithm

There are various existing machine-learning methods. We choose support vector machine (SVM) because of its powerful classification ability and well generalization ability [9]. 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: $\{\mathbf{feature}_i, level_i\}_{i=1}^{l}$, 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. CASE STUDY: ACADEMIC PAPER SEARCH

In this section, we demonstrate how to apply the proposed approach to a specific domain by searching for academic papers.

In our Libra [23] paper search system, about 0.9 million paper objects are crawled from the Web and indexed. About 0.4 million of them have full-text in HTML format (converted from PDF format via a converter). We collected citation information of each cited paper as its anchor text.

The process of extracting pseudo-anchor text for paper objects is exactly the process of implementing the following key operations in academic domain: pseudo-URL extraction, candidate anchor block extraction; candidate block accumulation, and pseudo-anchor extraction.

Pseudo-URL Extraction In the academic search domain, when one paper cites (or links to) another paper, a simple symbol (e.g. "[1]", "[5-8]") 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 reference in the references section a *reference item*. We locate the reference section by search for the last occurrence of term 'reference' or 'references' in larger fonts. Then, we adopt a rule-based approach to divide the reference 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 paper objects 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 object. We define the pseudo-URL for a paper object according to its title, authors, publisher, and publication year, because these four kinds of information can readily be used to identify a paper.

Candidate anchor block extraction Observing that most papers include the identity of the paper being cited in brackets, we locate reference points by examining each character sequence between '[' and ']' and look up the associated material in the reference section[4]. For each reference item, we treat the sentence containing its reference point as a candidate anchor block (another straightforward choice is to use the paragraph containing the reference point).

[4] There are of course a small percentage of exceptions. We choose to omit them for simplicity.

Candidate block accumulation All candidate blocks belonging to one paper object are accumulated via the feature-string hashing algorithm presented in Figure-6. We use term-level bigrams as feature strings (note that other kinds of feature strings were also tested). The similarity between two paper objects is computed as a linear combination of the similarities on the following fields: title, authors, venue (conference/journal name), and year.

Pseudo-anchor extraction We directly adopt the machine learning approach in Section 2 to assign weights for all terms in each anchor block. For training the SVM classifier, we labeled 2000 candidate blocks. Four importance levels are used in labeling the blocks. The features for training and testing are normalized versions of those in Table 3.

4. EXPERIMENTS

In this section, we verify the analysis of previous sections and test the performance of our approach in experiments. We demonstrate the effect of pseudo-anchor text in improving the performance of searching paper objects in academic domain.

4.1 Experimental Setup

All experiments are conducted based on Libra, our academic paper search system. We crawled and extracted 0.9 million paper objects from the Web, with nearly 0.4 million of them having full-text. The 0.4 million papers are processed according to the approach in Section 2 (and Section 3). In processing the references of a paper (say *A*, for example), if the paper being referenced (say *B*) is in our 0.9 million collection, we accumulate *B*'s candidate anchor block and increase *B*'s citation count by 1. Otherwise, this reference item is discarded. By this way, we get a rough citation count for each paper. All the 0.9 million papers (with their anchor text) are indexed (for the 0.4 million papers with full text, their title, authors, abstract, year, and full-text are indexed, while for the remaining 0.5 million papers, only the first four fields are indexed). A naïve word breaker (which treats characters other than letters and digitals as punctuations) is utilized to separate document text into terms. All terms (stop words and non-stop words) are indexed without stemming.

We randomly selected 300 queries from Libra's query log and sent them to researchers and students for selection in our organization. Each researcher or student was free to choose queries based on personal expertise, or propose new queries according to interest. Overall 4 researchers and 10 students participated in labeling, and 88 queries were labeled. The labeled queries are distributed in information retrieval, machine learning, system, database, and other fields in computer science. Here are some sample queries,

> *Link analysis, Parallel computing, Grid computing, Association rules, Peer-to-peer measurement, Collaborative filtering, parameter estimation, High dimensional indexing, minimum cut, …*

For each query, the top 30 results of all ranking algorithm were mixed and labeled by researchers and students. Each result is assigned a relevance value from 1 (meaning 'poor match') to 5 (meaning 'perfect match'). The reason for using five-judgment levels instead of the binary judgments widely used in information retrieval is that we believe multiple judgment levels can more precisely evaluate the relevance of an object to a query.

In labeling, we find that it is much more time-consuming to determine the relevance of a research paper to a query than the relevance of a Web page (with respect to a query). Although each researcher or student is in charge of labeling his/her familiar fields, there are nearly always some papers returned having never been read. In this instance, their abstract or even full text has to be examined before making decisions.

Since we used five judgment levels, some common evaluation metrics (e.g. mean average precision, precision@10, etc) are not applicable any more. In order to study the performance of our approach, we adopt the nDCG (normalized DCG) [8] measure in the experiments to evaluate search results. nDCG has two kinds of parameters: discount factor *b*, and gains for all labeled relevance levels. In our experiments, the value of the discount factor *b* is fixed to be 2. And the gain value for the 5 relevance levels (from 1 to 5) are 0.01, 1, 3, 7 and 15, respectively. For completeness, we also transform the 5 judgment levels into binary judgments (with judgment level 1 and 2 treated as irrelevant, and other levels as relevant), and utilize traditional IR evaluation metrics to evaluate our results.

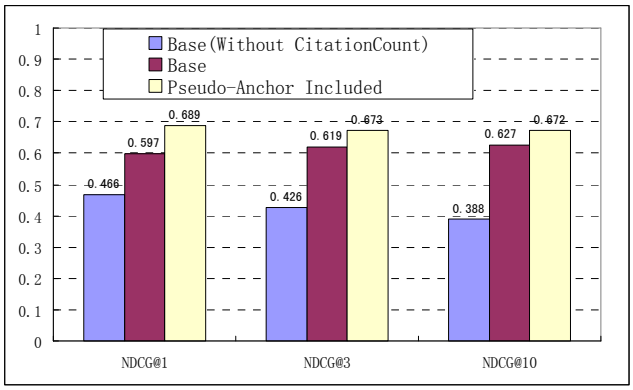


Figure 7. Comparison between the baseline approach and our approach (measure: nDCG)

4.2 Overall Effect of Our Approach

Figure 7 shows the performance comparison between the results of a baseline paper ranking algorithm and the results of including pseudo-anchor text in ranking.

The baseline algorithm considers the title, abstract, full-text and citation count of a paper. In a bit more detail, for each paper, we adopt the BM25 formula [1] over its title, abstract, and full-text respectively. And then the resulting score is linearly combined with the normalized citation count of the paper to get its final score. The normalization formula for citation count is as follows,

$$CitationCount_{norm} = \log(1 + CitationCount)$$

Please refer to the first paragraph of section 4.1 about how we collect citation counts.

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 primary ways of anchor score computation. The first is to merge all pieces of anchor excerpts (extracted in previous section) into a larger piece of anchor text, and use BM25 to compute its relevance score. In another approach called homogeneous evidence combination [30], 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,

$$S_{anchor} = \sum_{i=1}^{m} \frac{1}{(1+c\cdot(i-1))^2} \cdot s_i \qquad (2.1)$$

where s_i (i=1, …, m) are scores for the m anchor excerpts, and c is a parameter. The primary idea is to let larger scores to have relative greater weights. Please refer to [30] 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 7, we can see that overall performance is greatly improved by including pseudo-anchor information. Table 4 shows the results of t-test. A ">" indicates that the algorithm in the row outperforms that in the column with a p-value of 0.05 or less, while a ">>" means a p-value of 0.01 or less.

Table 4. Statistical significance tests (t-test over nDCG@3)

	Base	Base (without CitationCount)
Pseudp-Anchor Included	>	>>
Base		>>
Base (without CitationCount)		

Table 5 shows the performance comparison by using some traditional IR measures based on binary judgments. Since the results of not including CitationCount is much worse then the other two, we omit it in the table.

Table 5. Performance compassion using binary judgment measures

Measure / Approach	MAP	MRR	P@1	P@5	P@10
Base (including CitationCount)	0.364	0.727	0.613	0.547	0.501
Pseudo-Anchor included	0.381	0.734	0.625	0.570	0.531

4.3 Sample Query Analysis

Here we analyze one sample query to get some insights about why and how pseudo-anchor improves search performance. Figure 8 shows the top-3 results of one sample query: "TF-IDF". The results of Google Scholar are also listed for comparison. Google Scholar results are acquired on Jun 4th 2006, with the interface language and search language both set to be English in search preferences (this is important, because different results may be acquired with different preference settings).

For query "TF-IDF", both the baseline approach (without anchor) and Google scholar return some papers in which the TF-IDF weighting scheme is used or mentioned. Although the returned papers are relevance to the query, they are not excellent because users actually want to get the first TF-IDF paper or some papers introducing TF-IDF. When anchor information is utilized, some excellent results (B1, B2, B3) are generated. The main reason for this is that these papers (or books) are described with "TF-IDF" when some other paper cites 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

C1. [BOOK] T Joachims. **A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization**. 1996.
C2. DM Blei, AY Ng, MI Jordan. **Latent Dirichlet allocation**. Journal of Machine Learning Research. 2003.
C3. DL Hiemstra. **A Probabilistic Justification for Using TF-IDF Term Weighting in Information Retrieval**. International Journal on Digital Libraries. 2000.

(c) Google Scholar

Figure 8. Top-3 results for query TF-IDF

4.4 Candidate Anchor Block Accumulation Experiments

We have done some experiments to test the effectiveness and performance of the multiple feature-string hashing algorithm presented in Figure-6. The duplication detection quality of this algorithm is determined by the appropriate selection of feature strings. When feature strings are fixed, the slot size threshold can be used to tune the tradeoff between accuracy and performance.

Table 6. Slot distribution with different feature strings

Feature Strings / Slot Distribution	Unigrams	Bigrams	Trigrams	4-grams
# of Slots	$1.39*10^5$	$1.24*10^6$	$2.80*10^6$	$3.39*10^6$
# 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

We take all the paper objects extracted from PDF files as input to run the algorithm. Identical pseudo-URLs are first eliminated (therefore their candidate anchor blocks are merged) by adding all pseudo-URLs into a hash table. This pre-process step results in 1,458,039 distinct pseudo-URLs. We tested four kinds of feature strings all of which are generated from paper title: unigrams, bigrams, trigrams, and 4-grams. Table-6 shows the slot size distribution corresponding to each kind of feature strings. Table-7 is the performance comparison among different feature strings and slot size thresholds. It seems that bigrams achieve a good trade-off between accuracy and performance.

Table 7. Performance comparison between different feature strings and slot size thresholds

Feature Strings	Slot Size Threshold	Duplicated Objects Detected	Processing Time (seconds)
Unigrams	**5000**	529,717	119,739.0
	500	327,357	7,552.7
Bigrams	**500**	528,981	8,229.6
Trigrams	**Infinite**	518,564	8,420.4
	500	516,369	2,654.9
4-grams	**500**	482,299	1,138.2

5. RELATED WORK

Several categories of work are related to ours.

The most relevant work may be those use anchor text or their surrounding text for various Web information retrieval tasks. It was known in 1994 [2] that anchor text was useful to Web search. 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 [5] to automatically compile some lists of authoritative Web resources on a range of topics. An *anchor window* approach is proposed in [5] to extract implicit anchor texts. Following this work, anchor windows were considered in some other tasks [3][4][6][7]. Although we are inspired by these ideas, our work is different because vertical search objects have many different properties from Web pages. From the viewpoint of implicit anchor extraction techniques, our approach is different from the anchor window approach. Even from the viewpoint of solving the problem defined in 2.3.1, our method is still different, because we consider the information of other candidate blocks in extracting pseudo-anchor text from one candidate block. The anchor window approach is somewhat simpler and easy to implement than ours, while our method is more general and flexible. In our approach, the anchor text is not necessarily to be in a window.

There has been some contextual description related work in some vertical search domains. In the academic search domain, Citeseer [24][25][26] 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 image search, some descriptions of an image can be extracted from its surrounding text. And the information can be used to build an index for retrieving image objects. Anchor aggregation is not supported by most (if not all) image search engines.

6. CONCLUSION AND FUTURE WORK

In this paper, we propose to improve vertical search performance by utilizing pseudo-anchor information. As pseudo-URL and pseudo-anchor text can both be implicit in some vertical search domains, more effort must be made for pseudo-anchor extraction. We have proposed an approach for extracting pseudo-anchor information for vertical objects. Our machine-learning approach has proven successful in automatically extracting implicit anchor text. By applying our proposed framework to the academic search domain, we see a significant performance improvement over basic approaches.

The proposed machine-learning approach can also be applied to general Web search (as another choice in addition to the anchor window approach [5]) for extracting more anchor text for a Web page. We would like to exploit it in future work. Compared with general Web search, vertical object search engines usually support structured queries. We will also study how to use the pseudo-anchor technique to improve the performance of structured queries. Another important future work is testing our approach in more vertical search domains.

7. 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.

8. REFERENCES

[1] S.E. Robertson, S. Walker, and M. Beaulieu. *Okapi at TREC–7: automatic ad hoc, filtering, VLC and filtering tracks.* In Proceedings of TREC'99.

[2] O.A. McBryan. *GENVL and WWWW: Tools for taming the web.* In First International Conference on the World Wide Web, CERN, Geneva, Switzerland, May 1994.

[3] E. Amitay. *Using common hypertext links to identify the best phrasal description of target web documents.* In Proceedings of the SIGIR'98 Post Conference Workshop on Hypertext Information Retrieval for the Web, Melbourne, Australia, 1998.

[4] T. Haveliwala, A. Gionis, D. Klein, and P. Indyk. *Evaluating strategies for similarity search on the web.* In Proceedings of the 11th International World Wide Web Conference, 2002.

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

[6] B. Davison. *Topical locality in the web*. In Proceedings of SIGIR, 2000.

[7] G. Attardi, A. Gulli, and F. Sebastiani. *Theseus: categorization by context*. In Proceedings of the 8th International World Wide Web Conference, 1999.

[8] K. Jarvelin, and J. Kekalainen. *IR evaluation methods for retrieving highly relevant documents.* In 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2000. (pp. 41~48).

[9] C.J.C. Burges. *A tutorial on support vector machines for pattern recognition*. Data Mining and Knowledge Discovery, 2, 121-167, 1998.

[10] S. Brin. and L. Page. *The anatomy of a large-scale hypertextual web search engine.* In Proceedings 7th International World Wide Web Conference, 1998.

[11] I.P. Fellegi, and A.B. Sunter. *A Theory for Record Linkage*, Journal of the American Statistical Association, 64, (1969), 1183-1210.

[12] W. Cohen, and J. Richman. *Learning to match and cluster large high-dimensional Data Sets*. In Proc. of 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2002.

[13] A. McCallum, K. Nigam, and L. Ungar. *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.

[14] L. Parsons, E. Haque, H. Liu. *Subspace clustering for high dimensional data: a review*. SIGKDD Explorations 6(1): 90-105 (2004)

[15] A. Baxter, P. Christen, T. Churches. *A comparison of fast blocking methods for record linkage*. In ACM SIGKDD'03 Workshop on Data Cleaning, Record Linkage and Object consolidation. Washington DC; 2003.

[16] A. Broder, S. Glassman, M. Manasse, and G. Zweig. *Syntactic clustering of the Web*. In Proceedings of the Sixth International World Wide Web Conference, pp. 391-404, 1997.

[17] VI Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics - Doklady, 1966.

[18] M. Collins and Y. Singer. Unsupervised models for named entity classification. In Proc. of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, 1999.

[19] D.M. Bikel, and S. Miller. Schwartz, R., and Weischedel, R. 1997. Nymble: A high-performance learning name-finder. Proceedings of ANLP: 194–201.

[20] I. Muslea. Extraction Patterns for Information Extraction Tasks: A Survey. Proceedings of AAAI Workshop on Machine Learning for Information Extraction. Orlando, Florida, July 1999.

[21] M.E. Califf, and R. Mooney. Relational learning of pattern-match rules for information extraction. Proceedings of the National Conference on Artificial Intelligence. 1999

[22] D. Freitag. Information extraction from HTML: Application of a general learning approach. Proceedings of the 15th Conference on Artificial Intelligence (AAAI-98): 517–523.

[23] Z. Nie, Y. Zhang, J-R. Wen, and W-Y. Ma. Object Level Ranking Bringing Order to Web Objects. In Proceedings of the 14th international World Wide Web conference (WWW 2005), May 10-14, 2005, in Chiba, Japan.

[24] CiteSeer: http://citeseer.ist.psu.edu

[25] S. Lawrence, C. L. Giles, and K. Bollacker. Digital libraries and autonomous citation indexing. IEEE Computer, 32(6):67--71, 1999.

[26] C.L. Giles, K. Bollacker, and S. Lawrence. CiteSeer: An Automatic Citation Indexing System. In Proceedings of the 3rd ACM Conference on Digital Libraries (DL'98), pp 89-98, 1998.

[27] Z. Nie, F. Wu, J-R. Wen, and W-Y. Ma. Extracting Objects from the Web. ICDE 2006.

[28] J. Zhu, Z. Nie, J-R. Wen, B. Zhang, and W-Y. Ma. Simultaneous Record Detection and Attribute Labeling in Web Data Extraction. SIGKDD 2006.

[29] S. Yu, D. Cai, J-R. Wen and W-Y. Ma, Improving Pseudo-Relevance Feedback in Web Information Retrieval Using Web Page Segmentation. Proceeding of the Twelfth World Wide Web conference (WWW 2003), 11-18, Budapest, Hungary, May 2003.

[30] S. Shi, R. Song, and J-R Wen. Latent Additivity: Combining Homogeneous Evidence. Technique report, MSR-TR-2006-110, Microsoft Research, August 2006.