2008

Developing web crawlers for vertical search engines: a survey of the current research

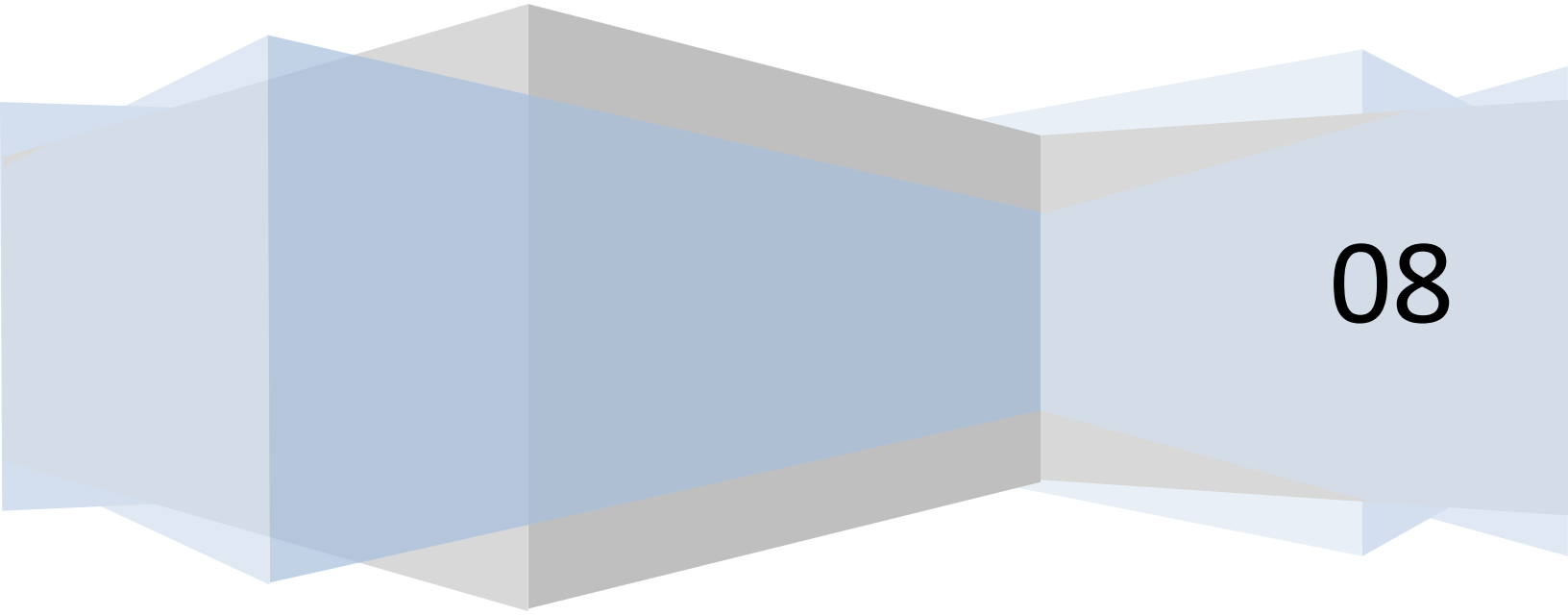
Pedro Huitema
Western Washington University

Computer Science Department – Western Washington University

Developing Web Crawlers for Vertical Search Engines

a survey of the current research

Pedro Huitema
Graduate Student
huitemp@cc.wwu.edu



08

Developing Web Crawlers for Vertical Search Engines

a survey of the current research

Pedro Huitema
Graduate Student
huitemp@cc.wwu.edu

Abstract

Vertical search engines allow users to query for information within a subset of documents relevant to a pre-determined topic (Chakrabarti, 1999). One challenging aspect to deploying a vertical search engine is building a Web crawler that distinguishes relevant documents from non-relevant documents. In this research, we describe and analyze various methods to crawl relevant documents for vertical search engines, and we examine ways to apply these methods to building a local search engine.

In a typical crawl cycle for a vertical search engine, the crawler grabs a URL from the URL frontier, downloads content from the URL, and determines the document's relevancy to the pre-defined topic. If the document is deemed relevant, it is indexed and its links are added to the URL frontier. Two questions are raised in this process: how do we judge a document's relevance, and how should we prioritize URLs in the frontier in order to reach the best documents first?

To determine the relevancy of a document, we may hold on to a set of pre-determined keywords that we attempt to match in a crawled document's content and metadata. Another possibility is to use relevance feedback, a mechanism where we train the crawler to spot relevant documents by feeding it training data.

In order to prioritize links within the URL frontier, we can use a breadth-first crawler where we just index pages one level at a time, bridges which are pages that aren't crawled but used to gather more links, reinforcement learning where the crawler is rewarded for reaching relevant pages, and decision trees where the priority given to a link depends on the quality of the parent page.

Keywords

Vertical Search Engine | Local Search Engine | Topical Crawling | Topical Relevancy

Introduction

Local search engines allow users to easily find Web pages relevant to a specific region or city. Suppose a user runs a query on Google for skiing; the first few results will point to online ski gear stores and general information on the topic. On most general search engines, to get locally-relevant results, the user would have to specify the name of the city or region as part of the query, and the engine would then ideally take that into consideration when returning results (Yu, 2007).

In contrast, if a user runs the same query on a local search engine, they will see results for ski areas around the region. The benefit of local search is that users may expect a higher level of precision in their query results, since queries are run against of subset of the World Wide Web, which has to be related to the engine's geographical region (Ahlers, 2008).

The largest difference between implementing a local search engine and a general search engine lies in how Web crawling is handled. A general search engine indexes everything it can find, and worries about determining the relevancy of those documents when a user runs a query. Local or vertical search engine needs to determine relevancy at indexing time in order to avoid adding unwanted documents to the collection. There are several approaches to writing a local crawler, which mostly revolve around how to identify a document as relevant or irrelevant (Gao, 2006).

It should also be noted that local search engines are a type of vertical search; therefore, techniques that have been used for other types of vertical search engines, such as medical search, could also work for local search engines (Ehrig, 2003).

In this paper, we describe and analyze a variety of methods used for crawling relevant Web pages for vertical search engines, and attempt to see how these would apply to a local search engine.

Methods

As mentioned above, a particularly challenging aspect of developing a local search engine is to efficiently index relevant documents.

The first major challenge is to determine whether or not a crawled document is relevant to the topic at hand. Most papers either present a keyword analysis type of method, where the parser looks for a variety of pre-defined keywords on the page and ranks a page as relevant if one or more of the keywords are present, or some sort of relevance feedback mechanism, where the parser has been trained with sample documents representing the ideal structure of what we are looking to crawl and compares those to incoming documents to determine relevance.

The second major challenge is to decide how the crawler should behave. There are many more approaches for doing this. The baseline approach, also known as breadth-first, is to simply index relevant documents one level of links at a time, and stop crawling when no more relevant documents have been found, or after reaching a certain amount of levels. Bridges are an extension to breadth-first, where we allow the crawler to continue crawling through a certain predetermined level of irrelevant documents in hopes of reaching relevant documents. Reinforcement learning is a machine learning technique where certain parameters are collected about documents, and the crawler is rewarded for finding relevant documents; the crawler then adjusts its crawling pattern to find more relevant documents first. With decision trees, the crawler prioritizes links to crawl based on how relevant their parent document(s) were.

Determining Relevance

Determining the relevance of a crawled document can be accomplished in one of two ways. First, we can use keyword analysis to check for predetermined words in the document and create a relevancy ranking. Second, we can use relevance feedback to compare crawled documents to training document and create a relevancy ranking based on their similarity.

Keyword analysis

This method focuses on determining whether or not a page is relevant. The administrator of the system provides sets of keywords that are deemed relevant to the topic (Badia, 2006) (de Assis, 2008), and each crawled document's raw text is analyzed to identify matches. The document's words can also be filtered through a dictionary to identify additional synonyms that could match the keywords.

Relevance Feedback

Tang determines the relevance of a given document by using training data to generate a list of relevant terms and phrases (Tang, 2005). This is done by parsing relevant documents and generating the most common terms and phrases from those documents. Tang then runs the same type of parsing, but this time against documents that are both deemed as relevant and of high quality, generating term and phrase lists that are a subset of the terms in the relevance analysis.

Crawling

For vertical search engines, the goal of a crawl is to come across more highly relevant documents before irrelevant document in order to minimize the amount of time spent indexing. Breadth-first crawling is our baseline method, where we index a pre-determined level of links from the seed pages. Bridges allow us to index links from non-relevant pages in hopes of finding clusters of relevant pages that could otherwise not be reached. Reinforcement learning rewards the crawler for finding highly relevant documents in hopes encourage it to pursue behaviors that lead to better crawling. Decision trees prioritize links to crawl based on the relevance of their parent page(s).

Breadth-first

Breadth-first crawling is the simplest method of crawling, and is used as a baseline for comparison by the papers consulted in this survey. Tang describes this method as traversing a link graph in a breadth-first fashion, and adding every newly discovered URL in a first in first out queue (Tang, 2005). Tang

expects the relevance of this crawler to fall as crawl progresses. Ahlers agrees with that statement (Ahlers, 2008), and mentions that a breadth-first strategy without any pruning would experience a rapidly growing list of URLs to crawl with an increased ratio of non-relevant pages.

Bridges

Ahlers explains that although in some vertical search engines, we can assume that topically related pages tend to link to each other, this is not always the case (Ahlers, 2008). In order to have a comprehensive crawl of, for instance, a local search engine, bridge pages need to be considered. As mentioned before, bridge pages are not relevant to the topic of the search engine, but may contain links to relevant pages, or more bridges that eventually lead to relevant pages.

A bridge page aware crawler needs to be given a radius (how many bridges to cross before giving up on a certain path). So when the crawler accesses a document, it needs to determine the relevancy of the document; then if the document is not relevant, consider how many bridges, if any have been crossed since we last accessed a relevant page on this path. If we haven't crossed the maximum number of bridges, then we continue crawling on this path; if we have crossed the maximum number of bridges, then we stop crawling on this path.

The theory behind using bridges is that if we only index relevant documents from a given starting point, we may be missing out on clusters of relevant documents that are indirectly linked from reachable relevant documents. We don't risk anything by making use of bridges, except for time, since irrelevant pages don't have to be indexed – their links are just followed.

Reinforcement learning

According to McCallum, the idea behind reinforcement learning is that a learner is given a task, represented by a set of states, a set of actions, a set-action transition function, and a reward function (McCallum, 1999). At each time step, the learner must perform an action, and then as a result receives a reward (or punishment), along with a new state. Eventually, the learner will develop a policy that maps states to actions in order to maximize rewards, and minimize punishments.

In order to apply this technique to crawling for a vertical search engine, McCallum recommends using the relevancy of the page containing a link, the relevancy of the link's anchor text, and the relevancy of the URL. Relevancy is determined by using a relevance feedback mechanism in this case.

McCallum uses this technique to locate computer science papers on University Web sites.

Decision tree

Li aims to use anchor text in links along with a decision tree in order to predict relevance for target pages (and prioritize crawling order) (Li, 2005). To do this, the crawler first needs to be trained with some example data. A graph of the training pages is generated, and the shortest paths from the entry page to each other pages are recorded. If an anchor text is on a short path, then it is used as a positive example; otherwise it is used as a negative example. Words from anchor text that only appeared in positive examples are kept in a positive repository, while words from anchor text that only appeared in negative examples are kept in a negative repository. The decision tree is then a function that returns

true if the anchor text has a majority of positive terms and false otherwise. Li uses this method to identify lecture Web pages at various universities.

Tang essentially uses the same method as Li, described above, but also includes words in the target URL and words in the 50 characters before and after the link (Tang, 2005). That research used decision trees to crawl documents that provide quality medical information.

Experimental Results

Before examining the experimental results of this current research, it is important to note that the experiments performed here had an identical goal of crawling relevant documents for a pre-determined topic, but that the collections that the documents were crawled from varied greatly. Some experiments, like Tang's attempt at finding highly relevant medical information, and Ahlers' attempt at identifying documents for a local search engine, were performed against the World Wide Web, an infinitely large collection. On the other hand, McCallum, for instance, crawls against a closed collection, which means that they would have eventually found all relevant documents, but less efficiently.

In addition, every research focused on a different topic for their vertical search engine, which means that a technique that works well for one topic might not work as well for another.

Because of this, we separate the experimental results into two distinct sections here; one which focuses on experiments against closed collections, and another which experiments on the entire World Wide Web. In addition, we will also list the topic that was being crawled for.

Closed Collections

We can immediately notice from Table 1 that in McCallum's experiments, breadth-first crawling does not give any advantage. Since the collection is closed, we are assured that every relevant document is indexed eventually, but there is no acceleration in terms of finding the most relevant documents first. Reinforcement learning, on the other hand, provides significant advantages, since their experiment was able to crawl 90% of relevant documents by looking at just 30% of the total collection.

Method	**Percent of links followed to retrieve x% of relevant documents**			**Topic**
	10%	**50%**	**90%**	
Breadth-first	10%	50%	90%	Research Papers
Reinforcement learning	2%	10%	30%	Research Papers
Decision tree	1%	5%	30%	University Lectures

Table 1: Crawling techniques against closed collections

Table 1 also shows that Li was able to use a decision tree method to reach 90% of the targeted documents after crawling just 30% of the total collection.

We can infer from this that reinforcement learning and decision trees act very similarly to find the most relevant documents first in a closed collection.

World Wide Web

Table 2 can be a bit misleading if we do not consider the topic being crawled for. First, to state the obvious, decision trees perform much better than breadth-first (2:1 ratio) when querying for medical information, as shown in Tang's research.

Crawling Method	% Relevance	Crawled Pages	Topic
Breadth-first	40%	~10,000	Medical Information
Decision tree	80%	~10,000	Medical Information
0 Bridges	30%	~7,000	Local
1 Bridges	38%	~10,000	Local
2 Bridges	17%	~182,000	Local
3 Bridges	10%	~190,000	Local

Table 2: Crawling techniques against the World Wide Web

Secondly, we see how changing the radius of bridges in Ahlers' experiments affects the relevancy (and number of crawled pages) of the resulting index. The 0 Bridges row presented in Table 2 is essentially the equivalent to performing a breadth-first crawl. When using just one bridge, the relevancy of the resulting index appears to be maximized; however, let us remember that this relevancy percentage is determined programmatically, not manually. This means that while using more bridges will force us to analyze more irrelevant documents, those documents will not need to be kept in our index, and ultimately, the total number of relevant documents is maximized (19,000 with three bridges, as opposed to 3,800 with no bridges).

Discussion

From the experimental results, it is clear that although breath-first crawling is a reliable method for reaching a specific amount of link levels from a starting document, and not running the risk of leaving out potentially relevant documents, it is by no means an efficient way to crawl for a vertical search engine. As the crawl goes deeper into a link structure, the chances of finding relevant documents quickly diminishes, and the crawl has to run longer in order to locate as many relevant documents as possible.

Reinforcement learning is a very efficient way to locate relevant documents as quickly as possible, because the crawler ends up following paths that maximize the chance of success. However, for this method to work there is an assumption that linkage between Web pages is consistent throughout the collection. This holds true in McCallum's experiments, as shown in Table 1, because there was a closed collection which consisted of university Web pages. The concern is that when launching such a crawler in an infinite collection (like the World Wide Web), the link structures will differ greatly, and this method could actually prevent us from reaching highly relevant pages which even a simple breadth-first crawler could locate.

Decision trees have been proven to work well in Li's experiments on a closed collection to find lecture notes (Table 1), as well as Tang's experiments on the World Wide Web to find medical information (Table 2). An area of concern with decision trees is that their effectiveness could be limited to the topic of the vertical search engine. In other words, while crawling medical documents related to a specific

condition could be very successful because they all share very similar terminology, crawling documents for a local search engine might not be as successful because the pages vary greatly in terms of the information that they provide.

Bridges, as presented by Ahlers' research, are essentially extensions to breadth-first crawling, but limit the depth of a crawl into a specified level of irrelevant documents (bridges). Table 2 shows that they are indeed capable of uncovering a great amount of relevant documents, while still requiring the crawler to filter through great volumes of irrelevant documents. For certain topics, especially like local search, bridges could very well be the method of choice.

Inspiration and Future Research Direction

My goal is to develop a local search engine for a specific city. Traditionally, geographic information is not readily available on Web pages, even if they are relevant to a specific city (Abou-Assaleh, 2007). This means that the crawler would need to be able to reach, and index local Web sites for shops, restaurants, services and general information, without necessarily finding references to the city on a given Web page. In a perfect system, a user could enter a search term and be presented with a list of links not only relevant to the term, but also related to the city.

One major difficulty in this area is that the World Wide Web is infinitely large, and pages relevant to a particular city do not necessarily link to one another. Therefore, even with a strong list of seed sites to crawl from, it will be very hard to retrieve every Web site that we would like to include in the search engine.

Another difficulty is to programmatically identify a Web site as relevant to a city or not. Vertical search engines that deal with general topics can easily generate training documents and keyword lists; however, in a local search engine, we resort to searching for mentions of the city or its landmarks in order to claim that a page is relevant, which means that relevant pages can easily be excluded, which irrelevant pages could be included.

One way to solve the issue of coming across as many relevant pages as possible is to involve the local community in the creation of the search index. I would like to see how, for instance, allowing business owners to submit their Web address to our crawler would help us increase coverage.

Since relevancy filtering is a bit weaker for local search engines than other vertical search engines, I would like to explore the possibility of using relevance filtering methods described here to build a listing of highly relevant seed pages, and then indexing the links on those pages without using relevance filtering.

Some good methods have been identified that could apply well to my research. First, in order to initially determine relevance for a given document, I would like to use keyword analysis with terms that are highly relevant to the city in question. Second, I was impressed with the idea of considering bridge pages while crawling, because the results were quite good, and the topic crawled was for a local search engine. This is behind the motivation mentioned above to build up a large index of relevant sites, and then let

the crawler loose for a few levels without checking relevancy, and finally turning relevancy filtering back on.

In the future, we will create successful local search engine solutions for various cities. The solutions will be generic enough that they will easily be applicable to other cities using different seed sites and relevancy inputs.

Bibliography

Abou-Assaleh, T. a. (2007). Geographic ranking for a local search engine. *In Proceedings of the 30th Annual international ACM SIGIR Conference on Research and Development in information Retrieval (Amsterdam, The Netherlands, July 23 - 27, 2007). SIGIR '07. ACM, New York, NY* , 911-911.

Ahlers, D. a. (2008). Urban web crawling. *In Proceedings of the First international Workshop on Location and the Web (Beijing, China, April 22 - 22, 2008). LOCWEB '08, vol. 300. ACM, New York, NY* , 25-32.

Badia, A. M. (2006). Focused crawling: experiences in a real world project. *In Proceedings of the 15th international Conference on World Wide Web (Edinburgh, Scotland, May 23 - 26, 2006). WWW '06. ACM, New York, NY* , 1043-1044.

Chakrabarti, S. v. (1999). Focused crawling: a new approach to topic-specific Web resource discovery. *In Proceedings of the Eighth international Conference on World Wide Web (Toronto, Canada). P. H. Enslow, Ed. Elsevier North-Holland, New York, NY* , 1623-1640.

de Assis, G. T. (2008). The impact of term selection in genre-aware focused crawling. *In Proceedings of the 2008 ACM Symposium on Applied Computing (Fortaleza, Ceara, Brazil, March 16 - 20, 2008). SAC '08. ACM, New York, NY* , 1158-1163.

Ehrig, M. a. (2003). Ontology-focused crawling of Web documents. *In Proceedings of the 2003 ACM Symposium on Applied Computing (Melbourne, Florida, March 09 - 12, 2003). SAC '03. ACM, New York, NY* , 1174-1178.

Gao, W. L. (2006). Geographically focused collaborative crawling. *In Proceedings of the 15th international Conference on World Wide Web (Edinburgh, Scotland, May 23 - 26, 2006). WWW '06. ACM, New York, NY* , 287-296.

Li, J. F. (2005). Focused crawling by exploiting anchor text using decision tree. *In Special interest Tracks and Posters of the 14th international Conference on World Wide Web (Chiba, Japan, May 10 - 14, 2005). WWW '05. ACM, New York, NY* , 1190-1191.

McCallum, A. N. (1999). Building domain-specific search engines with machine learning techniques. *In AAAI Spring Symposium on Intelligent Agents in Cyberspace 1999* .

Tang, T. T. (2005). Focused crawling for both topical relevance and quality of medical information. *In Proceedings of the 14th ACM international Conference on information and Knowledge Management (Bremen, Germany, October 31 - November 05, 2005). CIKM '05. ACM, New York, NY* , 147-154.

Yu, B. a. (2007). A query-aware document ranking method for geographic information retrieval. *In Proceedings of the 4th ACM Workshop on Geographical information Retrieval (Lisbon, Portugal, November 09 - 09, 2007). GIR '07. ACM, New York, NY* , 49-54.