The Topic Specific Search Engine

Benjamin Stopford

1st Jan 2006
Version 0.1 <DRAFT>

Overview

This paper presents a model for creating an accurate topic specific search engine through a focussed (vertical) crawling mechanism. The system is designed to run on limited physical resources (Most topics specific searches can be hosted using two well built home PCs). The implementation method is general but it is assumed that it would leverage from an open source search engine base such as Nutch.

The application is comprised of four parts; a crawler (spider), an indexer, a ranking mechanism (pre-emptive or lazy) and a user interface. The implementation presented here has the following extra features:

1. Crawling commences from a set of seed URLs. The seed set is generated by feeding positive and negative search keywords into a commercial search engine and using the resulting URL's as the seed list. The keyword set input into the search engine is defined by an expert user (this can be aided through data mining of positive and negative sample pages).
2. The crawling process does not cover the whole web. Instead it starts from the seed URL's created in (1) and crawls connected pages so long as they *continue to be considered relevant*.
3. Each page that is crawled is analysed for its relevance w.r.t. the search topic. This relevance value is based on both the initial keyword set and the relevance of the page's locality.

Synonyms

Topical search, vertical search, Vortals, focussed search.

Search Engine Basics

Search engines are typically consist of a crawler which retrieves pages from the internet. An indexer that registers web pages against the keywords they contain and finally a user interface that retrieves and ranks the results returned to the user based on their query.

Ranking Pages

The success of search engines tends to be attributed to their ability to rank the results. This is non-trivial as the average number of words in a user query is around two; hence the number of corresponding matching pages tends to be large (with the queryable web being several billion pages). The ability to rank pages sets of pages that all satisfy the user query is thus a fundamental attribute for success.

PageRank and Focussed Search

The success of Google can be largely attributed to the way it ranks pages. The ranking algorithm used, invented by Larry Page and Sergy Brinn is known as PageRank. The mathematical definition is quite complex and is unlikely to be of relevance to focussed search (see below) however the conceptual model they used is remains valid. The important feature of PageRank is that the rank of a page is increased by every page that links *to* it. Conceptually this is similar to a voting system (in fact the idea originally came from citations used to rank academic papers). When one page links to another, a portion of its PageRank is transferred to the page that is linked to. Thus the most linked to pages have the highest PageRank.

PageRank is important as it represents a method for ranking pages that is independent of the user’s query. This means that it can be performed off line. Unfortunately it is not applicable to focussed crawlers. The main reasons for this are that a focussed crawler does not have a complete view of the net, instead it sees only disparate pockets of relevant pages. This is discussed in more detail in [Cha03]. However this paper presents an alternative method of ranking that is only applicable to focussed crawling. This is a mechanism of ranking by topic relevance.

Indexing

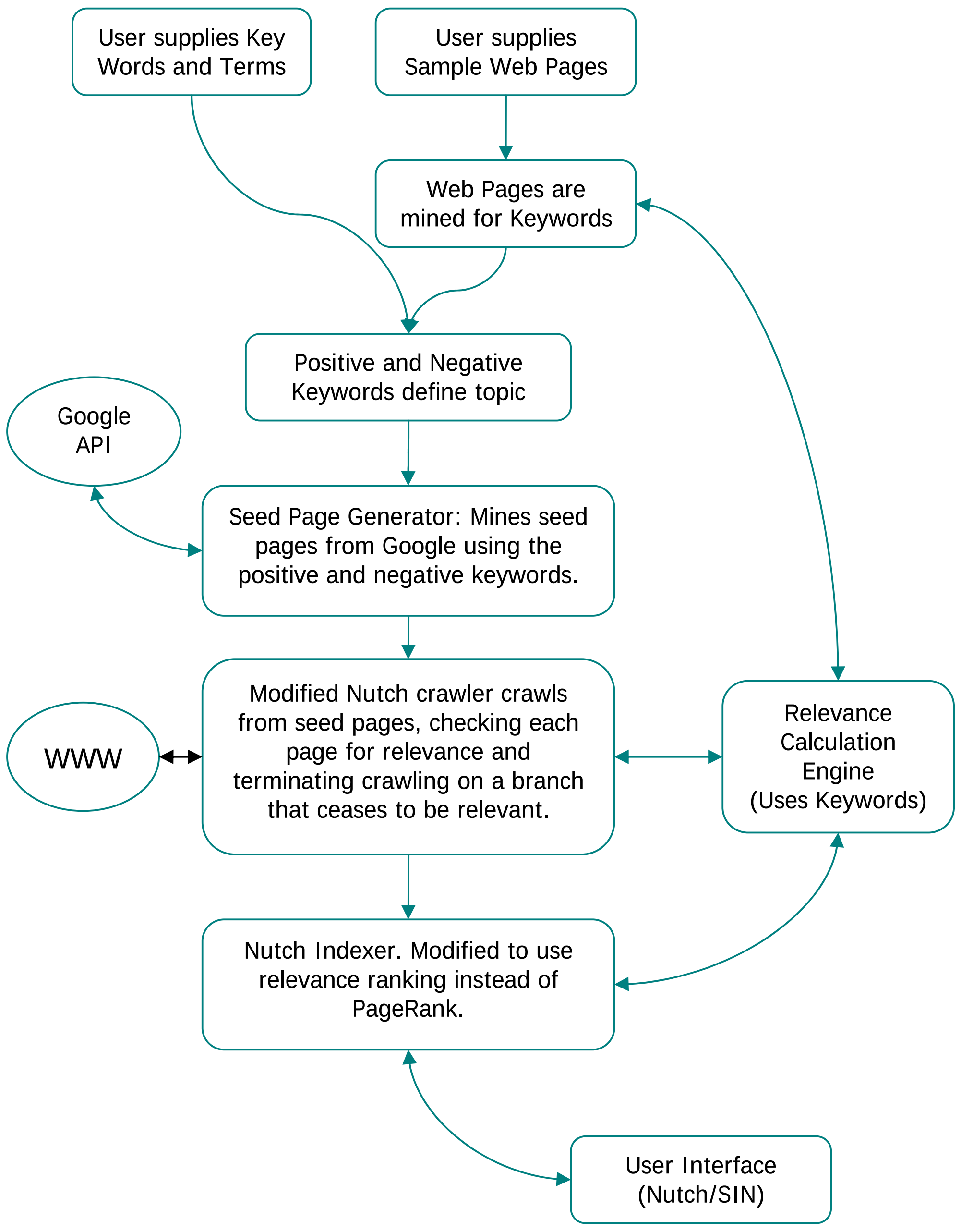
In general search engines do not use commercial databases to store the ranked pages. As the task is simple (store a look up of keywords to pages) a simple, custom data structure is usually used called an inverted file. An inverted file simply contains a key, the key word, and an attached list of page pointers. Multi word queries represent the intersection of the lists for each keyword. Each list is ordered by page rank.

Architecture

Crawler: Spiders the web retrieving pages. Crawlers generally keep local copies of the pages they crawl.
Indexer: This separate process takes crawled pages and adds them to the index. Indexers in most search engines run constantly but only put new indexes live periodically (every few days).
Retrieval and Ranking Server: The server must have sufficient resources to keep the inverted file index in memory.

The Topic Specific Search Engine

Summary of Architecture


User supplies Key Words and Terms
User supplies Sample Web Pages
Web Pages are mined for Keywords
Positive and Negative Keywords define topic
Google API
Seed Page Generator: Mines seed pages from Google using the positive and negative keywords.
WWW
Modified Nutch crawler crawls from seed pages, checking each page for relevance and terminating crawling on a branch that ceases to be relevant.
Relevance Calculation Engine (Uses Keywords)
Nutch Indexer. Modified to use relevance ranking instead of PageRank.
User Interface (Nutch/SIN)

Key Word/Phrase Set

The user of the system must define the search topic through a set of positive, negative terms that define the search engine topic. Positive/negative terms signify an increased/decreased likelihood of a page containing them being relevant. The existence of a Defining keywords or phrase in a page denotes that the page must be on topic.

Each of these sets is sorted in order of importance as in the below figure, and these ranked terms are used by the crawler to determine if a page is relevant.

Positive Defining Terms	Positive Term	Negative Term
VLCC	Shipping	FedEx
Dead Weight	Freight	Postage
...	Deadweight	Packing
	...	...

↑ Order of Importance

The keywords and phrases can either be created by an expert user or alternatively, they can be mined from a set of relevant and irrelevant sample web pages. Should they be mined it is advisable to have them reviewed by and expert user after the mining process has completed.

Mining Key Words

To mine keywords, instead of presenting known words and phrases, the user supplies a set of web pages that are known to be of relevance to the search topic. A negative set of specifically irrelevant pages are also provided as is a control sample random pages. These positive and negative sets would likely be constructed through an expertly user searching using a main stream search engine and determining the relevance of the results by hand.

The positive and negative pages are then turned into vectors in keyword space. Vectors are used as they provide a simple method for measuring, and comparing, both the existence and number of keywords in a document. The representation as vector however is an implementation detail, the important concept being that the documents are compared to find the words that most characterise the positive and negative sets. Positive and negative and defining terms are then mined from the document sets. The positive and negative terms are defined leading and trailing edge of the separation between keyword vectors. Essential terms are those that only appear in the positive document samples and not the negative ones.

As a final phase the essential, positive and negative terms are reviewed by an expert user. The expert user then fine tunes the ranking of the keywords w.r.t each other and removes any that are incorrect (due to the finite nature of the sample set it is likely that the positive documents will contain words not in the negative documents or the control. These can then be weeded out by an expert user.

The keyword set is expanded using latent semantic analysis. This removes issues arising from polysemy and synonymy.

Crawling Features

Searching starts from a seed URL. Searching is performed in a breadth first manor (all links from a page are followed before and secondary links from subsequent pages are followed). Documents whose relevance falls below the threshold level are no longer followed (in fact the best algorithm would be to have an overshoot function that allows a defined number of irrelevant pages to be crawled before terminating the branch (note that these irrelevant pages would be crawled but not indexed). When all paths from the seed URL's have been terminated by this mechanism crawling proceeds with a new seed URL. This mechanism is similar to the approach termed the Hopfield Net Spider [Cha03].

Calculating Document Relevance

During crawling the spider determines the relevance of each page w.r.t. the search topic to decide whether to continue down that particular branch of the crawl tree. The relevance of a page is determined via two factors:

- The similarity between the page and the keyword set.
- The relevance density of the surrounding pages (i.e. irrelevant pages that are in an area of high relevance inherit relevance by association).
- The average relevance of the site (if the site is highly relevant then it is worth crawling through irrelevant pages).

Using the Keyword set to find Seed Pages

The search engine crawls from a set of pages. These seed pages are found by feeding the keyword set into a commercial search engine. The Google API is suggested which is limited to 1000 searches a day, each yielding 10 results. Positive and negative search criteria can be used to query the search engine e.g. "Shipping -FedEx". The queries submitted to the Google API would start with *all* positive keywords, reducing the number of keywords incrementally but ensuring all combinations are used. For example; say that the dictionary contains the keywords A, B, C, the following queries would be submitted to Google to generate seed URL's:

1. A, B, C.
2. A, B.
3. A, C.
4. B, C.
5. A.
6. B.
7. C.

This method ensures that pages with the highest number of keywords will be crawled first.

Document Relevance as a means of Ranking

If the search engine user specifies an unusual combination of words then the retrieval task is simple as the number of matching pages will be small. However the more usual case is that the user types in a common search phrase resulting in numerous keyword matches. The question is how to order the results. As discussed previously the PageRank method favoured by Google will not work as the pages are crawled in a disconnected manor. Thus an alternative ranking mechanism must be used. The use of the relevance algorithm (with slightly different parameters to that used during crawling) is suggested as a method for rating the importance of each page. The algorithm differs to that used in crawling as it focuses on an absolute measure of relevance as apposed to an estimate of whether the page is likely to yield further relevant pages if crawled. In addition the standard search engine rating criteria are used. Namely:

- TF*IDF where TF = Keyword Frequency, IDF = Inverse Document Frequency [FOA].
- HTML attributes of the keyword term (heading text etc) [FOA].

Additional Features

- Types of pages. It is possible to categorise the type of page. Is it a company we site, research paper, resource page, shop, other.
- "Two vertical searches…" paper has details of an interesting page text summariser.
- Expand relevance analysis to incorporate **phrases** rather than just words.
- Increase the crawling efficiency by using analysis to determine which links to follow (although I'm not totally convinced of the worth of this. See Computer article).
- Clustering of results. Nutch supplied a plug-in for this.

Reuse of Existing Implementations

There are a variety of open source search engines available. Nutch, which is based on lucent technology, is probably the best and is available under an Apache license. This provides a full search engine service, written in java and would form a suitable base point for this project.

Project Phases

There are two prerequisite tasks that should be performed prior to starting the project. Firstly is a feasibility study into the proposed implementation. This should include analysis of the technical solution w.r.t. risk factors that pertain to its construction. In addition the feasibility study should include research into commercial off the shelf products that provide similar results.

The second prerequisite is a prototype system. This would involve creating a straight thorough implementation of the main components completing within half the project duration. The prototype system allows confirmation of the feasibility of the technology as well as providing a mile stone at which management can re-evaluate. The keyword set for the prototype will be created by hand by an expert user.

The project itself can be divided into the following phases (with very approximate timings):

- Installation of a sample Nutch search engine (assuming this is the chosen base implementation).
- Evaluation of the extensibility of Nutch for the specific customisations requirement for vertical search.
- Evaluation of Google API for applicability as the seed set generation mechanism.
- Create seed page generation application using the Google API (schedule late in project).
- Create sample page analysis tool to aid the creation of base keywords (schedule late in project).
- Modify Nutch so that it crawls from the set of base pages.
- Modify Nutch so that calls for relevance analysis on each page it crawls and halts should it go below a threshold value (start with mock implementation).
- Modify Nutch so that it incorporates a call to the relevance engine instead of using ranking factors such as PageRank.
- Write the relevance analysis tool which rates each page and connect to Nutch interface.
- Incorporate this relevance level into the Nutch framework for indexing and ranking.
- Customise look and feel.

Hardware: Nutch can support 20 million pages at average 1 query per second on a single node. Such a machine will need around 200 GB of disk space. The engine will support much higher capacity than this with hardware upgrade. A standard Linux workstation will run 200 fetchers fetching at 5 MB/s

References

[Cha03] Comparison of three vertical search spiders, Chau – Computer Magazine 2003
[FOA] Finding Out About – A Cognitive Perspective on Search Engine Technology and the WWW – Rickard Belew

Related Resources

Examples:
http://www.researchindex.com/
http://www.business.com/
http://searchenginewatch.com/links/article.php/2156351
http://www.searchengineguide.com/searchengines.html
http://scholar.google.com/

Articles:
http://searchenginewatch.com/sereport/article.php/2162541 (2000)
http://www.cioupdate.com/trends/article.php/3525311 (2005)
Focused Search
Scoring Systems for Vertical Search
Case Study of two approaches to Vertical Search

Commercial Alternatives:

http://www.vortaloptics.com
http://www.vortalbuilding.com/applications.html (A company that builds Vortals)

Open Source search engines:
http://www.lucene.com
http://lucene.apache.org/nutch/

Examples of Nutch in action being:
http://askaboutoil.com/ (focused search based on Nutch)
http://www.mozdex.com/ (uses Nutch clustering plug-in to cluster results)
http://java-source.net/open-source/search-engines