1 Web Search Engine

My source for this part about web search engine is book by Langville and Meyer [1].

1.1 Main functionality

Web search engine can be simply described as a tool for information retrieval from world's largest linked document collection known as world wide web. Compared to traditional information retrieval (for example in library) which search in non-linked collections that are sorted by many aspects, the web information retrieval search through linked pages with no straightforward categorization. On the contrary, the web is self-organized by links from one page to another page.
The search engine consists of several elements that allow us to find the most relevant information in the linked collection of information.

1.2 Structure of Search Engine

The search engine can be divided to submodules. These submodules are either query-independent or query-depended. Query-independent modules work on their own without any user input. Query-depended modules work in real-time and are initiated when user starts a query.

1.2.1 query-independed modules

Crawler Module As was said in the beginning the web is not a central collection with any categorization as a traditional library. It can grow without any geographical limit and without any categorization. The search engine, therefore, has to do the categorization on its own. This categorization is done by crawler module which collects and categorizes the web's documents. The crawler module uses a software called spider which continuously gather new information about web pages.

Data/Page Repository The complete information about web page gathered by spiders is then sent to page repository where it is temporarily stored until it is sent to indexing module.

Indexing module There the page information is stripped off of unimportant parts and compressed to make small entry in Indexes. The full page is then tossed away or, if it is frequently used, returned to the page (data) repository.

Indexes There are many special indexes that hold particular compressed information about web pages. Few examples are content indexes, structure indexes and special-purpose indexes.

1.2.2 query-depended modules

Query module It converts query input of user (usually natural language) to information readable by search engine. It then search in indexes for the relevant pages which are then ranked by ranking module.

Ranking module This module sorts the list of relevant pages so that the pages which user most likely want are on top of search and the irrelevant ones are on bottom of list. It usually combine two scores of pages called content and popularity score.
Content score is calculated based on many rules. Pages with query words in their title are usually awarded with more points then pages with query words only in their body.
Popularity score of page is based on the hyperlink structure of Web.
One example of ranking module is PageRank used by Google search engine.

2 Crawlers

We now know how simplified search engine's structure looks like. Let us inspect the crawlers in more detail. My source for this part was chapter 8 in book Web Data Mining [2].

2.1 Crawling mechanism

The Web can be seen as a graph where nodes are pages and edges are links between them. The crawler starts from set of seed pages and uses links to get to other pages. This simple graph search has many issues which are hard to solve. Web crawler is the most sophisticated yet fragile component of a search engine as is said in [3]. We can search through the graph of the Web using one of many approaches but the usual ones are based on the breadth-first search.

2.2 Crawling strategies

The crawling strategies which are based on breadth-first search maintain list of unvisited URLs called the frontier. This list is initialized with seed pages. In each iteration we get the first page in the frontier and process the page.

2.2.1 Breadth-first Crawler

The ordinary breadth-first crawler implements the frontier as a FIFO queue.
We would normally assume that the pages are visited in random order, but that is not the case with the Breadth-first crawler. As the links pointing to pages are not distributed normally around the mean indegree, rather there are some pages that have indegree exponentially larger than the mean indegree. This causes the crawler to be biased towards visiting the well connected pages.
Another reason which affect the crawler preference towards certain pages is the choice of seed pages. The pages that are topically similar to the seed pages are usually much more likely to be related to the seed pages than randomly selected pages.

2.2.2 Best-first (Preferential) Crawler

The Best-first crawler uses priority queue as its frontier instead of FIFO queue as in Breadth-first crawlers case. A priority is assigned to each unvisited link in the frontier based on combination of many different properties. These properties can be for example of topological type (for example indegree of the linked page), or content type (for example count of query words occurrence on the page).

2.3 Characteristic properties of Crawlers Types

Crawlers can be divided not only by the type of the crawling strategy, but also by what they are used for. There are three main types which we will discuss. These are Universal, Preferential-focused, Topical, and Intelligent crawlers.

2.3.1 Universal Crawler

These are large scale crawlers that process hundreds of thousands of pages per second to maintain indices of the most important pages as fresh as possible. The importance of pages is based on various factors such as popularity measure. PageRank for example computes importance of current page based on the links pointing to it and also based on the page rank in previous iteration of these pages from which the links are pointing to the current page.
As is said in the book [2], the major commercial search engines had size of index around 10^{10} pages at the time of the book release.
There is conflict between the main goals of universal crawler which is to cover as many pages as possible and simultaneously maintain the index as fresh as possible. The problem of right balancing of these two goals is source of many studies as is shown in the section 8.3.2 of the book.

2.3.2 Preferential-focused Crawler

These crawlers attempt to be biased towards certain category of pages that interest user. Like in the case of universal crawler, the pages are assigned a rank. But this rank is not related to the indegree of the page, but rather it is a probability that the page belongs to the certain category. It can be computed for example by Bayesian classifier.

One category of focuesd crawlers are Context-Focused Crawlers which from seed page create a context graph with L layers and then assign a probability to each layer. Then in the time of crawling, this probability is used as as threshold to decide if the page is relevant or not.

2.3.3 Topical Crawler

In case when we do not have a sufficient number of pages to train the classifier as in the case of focused crawler, we can start with only several seed pages and the description of topic. This type of preferential crawler is called topical crawler.
Another characteristic of topical crawler is that it visit (crawl through) pages in real time so they all have fresh score. This can be used to find very recently linked (or created) web pages. The disadvantage is that querying is much slower than in case of universal crawler with already indexed pages.
In case of the topical crawler we can use some cues to establish higher priority of some pages over others. We can use so called link topology.

2.3.4 Intelligent Crawler

All the crawlers described until cannot adapt to certain topics or context of the pages. They use static strategy to evaluate unvisited pages and to manage the frontier. This strategy does not change in time of crawl.
There are also the Intelligent crawlers that use statistical models to learn how high priority should a certain page in frontier have.
These crawlers use certain types of machine learning and reinforcement learning which will be discussed in the next section.

2.4 Use of Reinforcement Learning in Intelligent Crawlers

The problem of crawling through web pages can be defined as a reinforcement learning problem. Crawler can be an agent which tries to maximize his future reward discounted over time. As is described in the [2], pages are in this sense the states ($p \in P$) and crawler agent has function $L : P \times A \to P$ using links as an action ($a \in A$) to get from one page to next page. Each movement is rewarded with reward function R like this: $r : P \times A \to R$. We want to find a policy π which maps the states to actions $\pi : P \to A$ and maximizes future reward discounted (by factor $0 \leq \gamma < 1$) over time: $V^{\pi}(p_0) = \sum_{t=0}^{\inf} \gamma^t r(p_t, a_t)$. We then define a function which returns the value of selecting action a from state p: $Q(p, a) = r(p, a) + \gamma V^*[L(p, a)]$. The V^* is the value function of the optimal policy $\pi^*(p) = argmax_a Q(p, a)$.

2.5 Evaluation of Crawler's Performance

We can again use a classifier as in the case of preferential-focused crawler. But this time the classifier has to be trained on different data than the crawler which will be evaluated so that the evaluation is unbiased.
We can then assess the crawlers performance using metrics precision and recall.

2.5.1 Precision

For precision, we can again use the classifier with threshold probability. The pages with probability that they are relevant higher than threshold are classified as good and pages with lower probability are classified as bad. We can then compute precision with use of number of good and bad pages.

2.5.2 Recall

Crawler can be also evaluated in terms of search length. Search length can be defined as the number of pages crawled before a certain percentage of the relevant pages is found. Recall is the percentage of documents found and identified as relevant compared to the total number of existing documents.

2.5.3 The Role of an Appropriate Choice of Seed Pages

If we have an experiment with a set of known relevant target pages and want to measure performance of a topical crawlers, we cannot use these target pages as a seed pages. We therefore have to come up with strategy to obtain seed pages that differ from the target pages.

2.5.4 Viable Strategy to Obtain Seed Pages

There are two strategies described in the book [2]. First strategy performs a back-crawl from the target pages until the link distance from target pages is far enough so that the process of finding these target pages is hard enough for the crawlers being compared.
Another approach is to split the set of known relevant pages to targets and seeds. It is much more simple than the back-crawl as we don't need to do any crawling and just split the set. Disadvantage is that it is not guaranteed that the targets are reachable from seed pages.

Bibliography

[1] Amy N. Langville and Carl D. Meyer: Google's PageRank and Beyond: The Science of Search Engine Rankings, Princeton University Press, 2006.

[2] Filippo Menczer: Web Crawling (Chapter 8), in: Bing Liu: Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data, Springer, 2008, pp. 311-362.

[3] S. Brin and P. Lawrence. The anatomy of a large-scale hypertextual web search engine.Computer Networks, 1998, 30(1-7): p. 107-117.