INTELLECTUAL SEARCH ENGINE OF ADEQUATE INFORMATION IN INTERNET FOR CREATING DATABASES AND KNOWLEDGE BASES

Oleksandr Kuzomin, Bohdan Tkachenko

Abstract: *Modern Web is more comparable to boundless universe of information. Sometimes it is very difficult to find relevant information in Web because of a variety of information and "unfair play" by some of webmasters. Even popular and powerful search engines like Google, Yahoo, Yandex and Bing sometimes cannot provide relevant information. In this work we are trying to find new methods, algorithms and approaches to make Web search faster and more efficient.*

Keywords: *information retrieval, content analysis, document indexing, information storage, search engine, crawler*

ACM Classification Keywords*: D.2.11 Software Architectures, H.3.1 Content Analysis and Indexing, H.3.2 Information Storage, H.3.3 Information Search and Retrieval*

Introduction

We created cluster for informational retrieval and analysis. It collects information across the Web and stored in our database. It can collect information in parallel with unlimited number of nodes, so it is easy to scale crawler. Then it processes collected information and creates knowledge base. It determines keywords and entities from text of each and every document, as well as dates, addresses and names. It also stores all the previous versions of documents, so you can search them even when document is not more available in Web. All this information helps to distinguish relevant and irrelevant information.

Crawling Web and Storing Documents

To make crawlers really independent and fast we used the following approach: Crawler asks from Queue for next URL and then Crawler downloads this URL and puts it to Documents Storage. Then it just asks for the next URL. When new URL appears at Documents Storage, initial processing is started [Miner, 2012]. Initial processing extracts all URLs from the document and puts new ones in the Queue, so Crawlers later will download these URLs too. So this approach allows us to download as many documents as possible independently. This algorithm is shown on Figure 2.

```
var redis = require('redis');
var request = require('request');
var client = redis.createClient();
var storage = require('./storage_client');

function getNextURL() {
    client.brpop(['queue', 0], function (listName, url) {
        request(url).then(function (data) {
            storage.put(url, new Date, data);
        });
    });
}

getNextURL();
```

Figure 1:

Queue is implemented on basis of Redis. Redis is a very fast and reliable key-value store and it is very useful in our case. Below is a part of Crawler's code (Figure 1).

There is also another interesting part - Documents Storage. It is not just a URL -> Document storage. It also has versioning features, so it can store different versions of documents. It is actually stores just differences between documents, so stored documents need less storage and, as a result, we can store more documents on our servers. We implemented a special version and document-oriented. It uses minimum amount of storage and in the same time it can rapidly deliver requested document.

Initial processing step is quite easy. It extracts all URLs with simple regular expression query (Figure 3).

Indexing documents and making Knowledge base

Old search engines were very slow because every time when user was making query, search engine searched this text in each and every document. But modern search engines are making indexes when document appears in the system. So we are using the same approach here. When new document appears in our Documents Storage, independent subsystem, called Indexer is processing these documents, fetching different data and metadata from it.

Basically, it splits document by words and makes index for each and every word in a document: later, when we need to find some word in our Documents Storage we can just look up in Index Storage for this word and quickly find ALL documents with this word with the positions where this word can be found in the document. It is enough to find documents, but we need more information to rank information and return only relevant information for user.

So, on this step we also collect additional information:

1. Distance between each word in a document

When user makes query like "cheap flights to Germany" he means that he wants to take a cheap flight to any city in Germany. If we will just show him pages that have words "cheap", "flights", "Germany" then we cannot guarantee that first results will be relevant, because there are a lot of pages that have these words, but they are scattered along

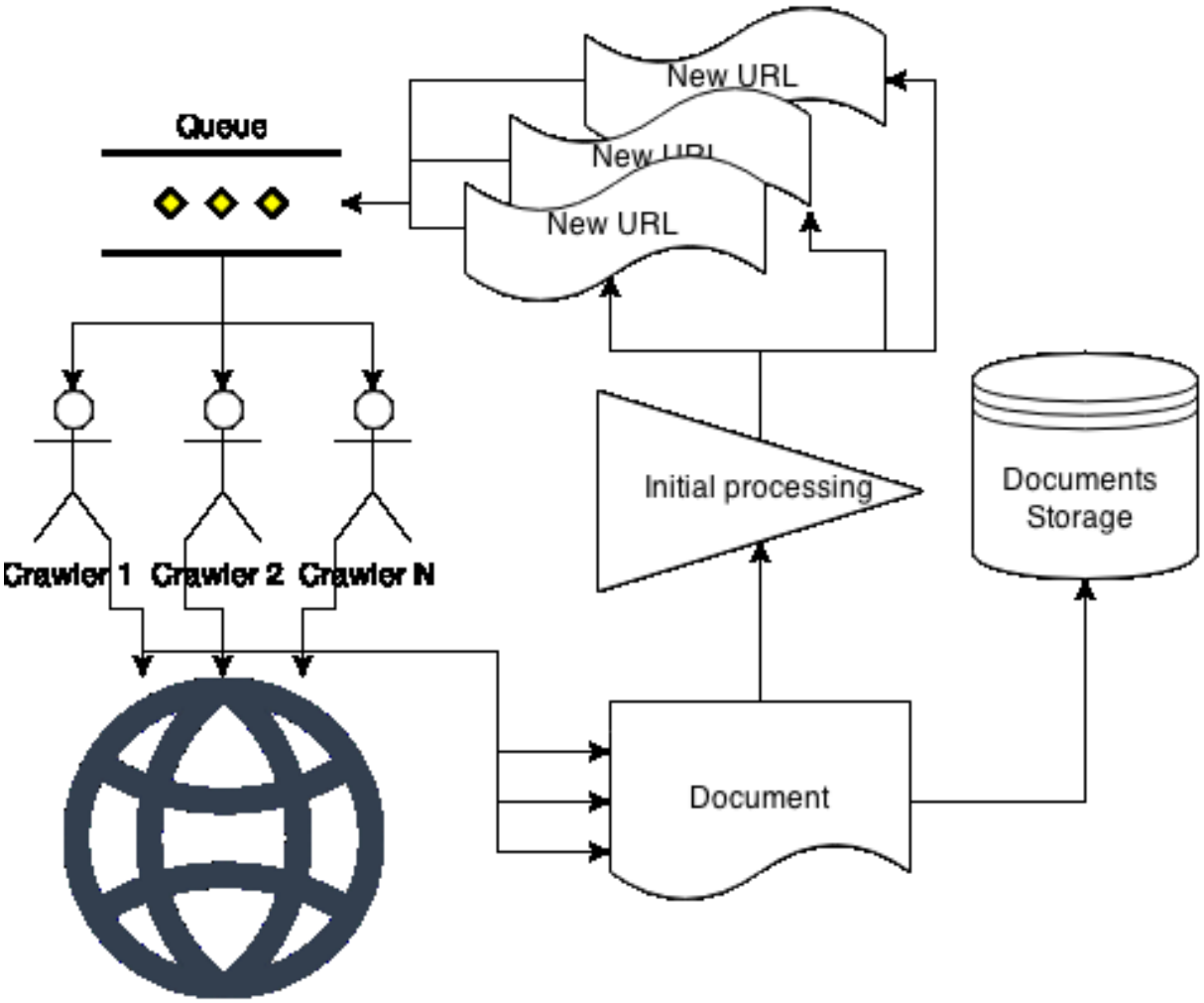

Figure 2:

```
/(http|ftp|https):\/\/[\w\-_]+(\.[\w\-_]+)+([\w\-\.,@?^=%&amp;:/~\+#]*[\w\-\@?^=%&amp;/~\+#])?/g
```

Figure 3:

Figure 4:

the document. In this situation we need to know a distance between all words in document, so we can first show documents where is words are close to each other.

2. Context

Context is several words before and after each word. We need context for each word for different purposes: 1) when showing results of search we also want to show a context, so it is much simpler for user to understand if it is relevant result for him; 2) we want to extract phrases which include this words. It is useful for some queries where we can have "perfect match" of users query.

3. Keywords

Keywords are very important because they can show the main idea of a document. Often keywords can already be found in page metadata as webmasters and SEO professionals put them there [Manning, 2008]. But we also parsing document to find the most frequent used words in document. Then we filter simple words to get only list of terms. Each keyword has its own weight. Weight is bigger if keyword appears more frequent. Visual representation of keywords weight is shown on Figure 4.

4. Entities

We are checking each keyword in our knowledge base to find according entity. Entities are not plain words, they represent words in the same way as human understands them. When we are talking about some entity we assume, for example, a color, a material or a substance of this entity. If it is country then we know it's size, population and history dates [Takhirov, 2012]. All this information is very useful in our search. When user queries about some entity and some entity property then we can even give him answer right above the search results. For example, for query "what is the population of Ukraine" we can detect that we have entity "Ukraine" and this entity is "country" and

country has property "population" and we also have exact property value in our Knowledge Base, so we already can answer user "45 million".

5. Author

It is also very important, because when we know author of a document, then we can first show, for example, documents with favorite author of a user or to show documents with researchers in a zone of interest of a user.

There are several ways to detect an author of a document:

- Document metadata (in head of HTML code)
  <meta name="author" content="Bill Gates" />
- Direct match in text
  Author: Bill Gates
  Published on ... by Bill Gates
- Indirect match in text
  My name is Bill Gates and I want to tell you a story about...
- Quotes in other documents
  Citations: "You must be the change you wish to see in the world. - Gandhi"
  URLs to document: "New article by Bill Gates - http://..."
- Social networks
  For example, someone wrote an article and said in Twitter "My new article about search engines - http://..."

Of course, some of these methods are note very reliable, but in sum several methods can give pretty good results.

6. Geolocation

Very often users want to get local result first. For example, if user is making query like "pizza" then probably he wants to see pizza restaurants nearby. So we need to detect a location of every document and then rank result by distance to the user (Figure 5). We can do this by detecting IP address of a resource and then using GeoIP database we can detect the location. But, this location will not be very accurate. But, luckily, websites like pizza restaurants often have exact address information on a separate page or in the header of the page. So we can use this information to get most accurate location of a resource.

7. Zone of interest

We are trying to sort each document by zones of interest and save this information in metadata. So when our user is photographer then probably he wants photography and art-related results, but when user is engineer then on the first place he wants results that are related to engineering [Gauch, 2003].

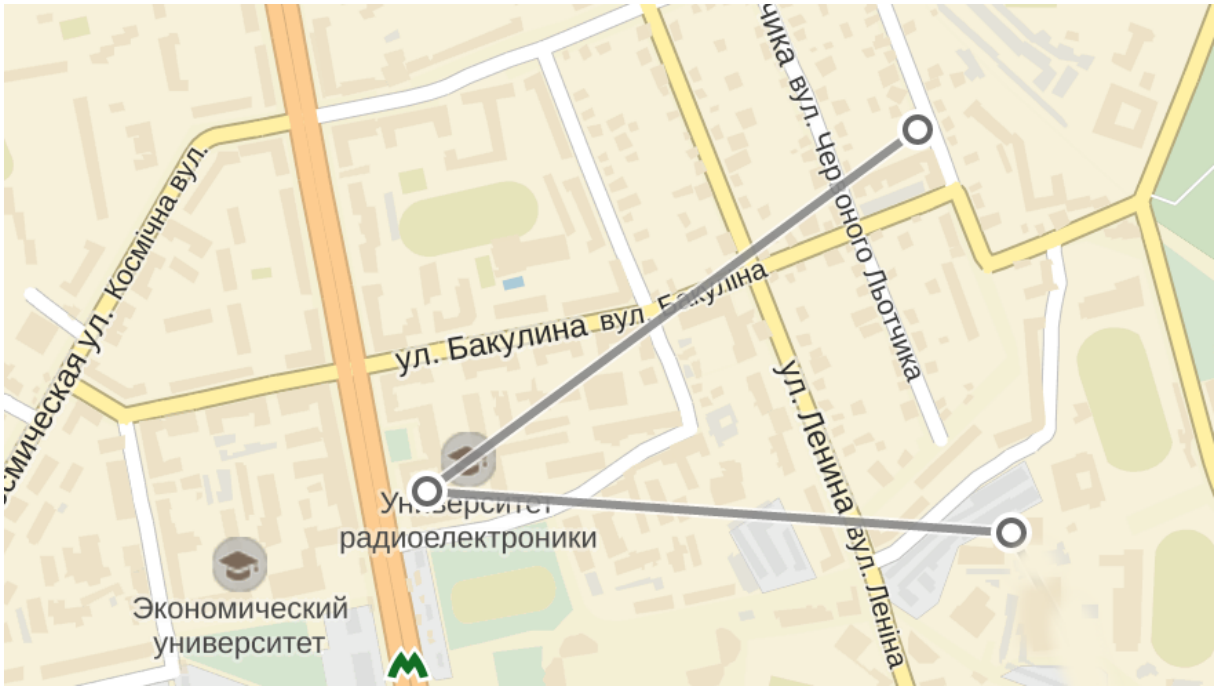

Figure 5:

To detect zone of interest of user we are recording all his searches to make prediction of his zone of interest. We are also using his profiles in social networks and his public messages in social networks.

To detect zone of interest of a document, we are using top keywords and entities of a document to find best zone of interest of this document.

On Figure 6 we can see results for sample document. Each blue dot is a keyword. They appear on a plot by their usage frequency and by distance to each other. So we can select three groups of keywords and assume that these three groups are zones of interest. Next we can try to find existing zone of interest with such terms and if such zone is not exists then we can create new one.

Improving the knowledge base

We need to have frequently updated and fact-oriented knowledge base to detect entities in our document. Luckily there are several existing open knowledge bases. We are using them as the basis for our knowledge base and also we are expanding this knowledge by knowledge that we got from Web. Currently we are using following knowledge bases:

- Wikipedia
- DBpedia
- Freebase
- Wikidata

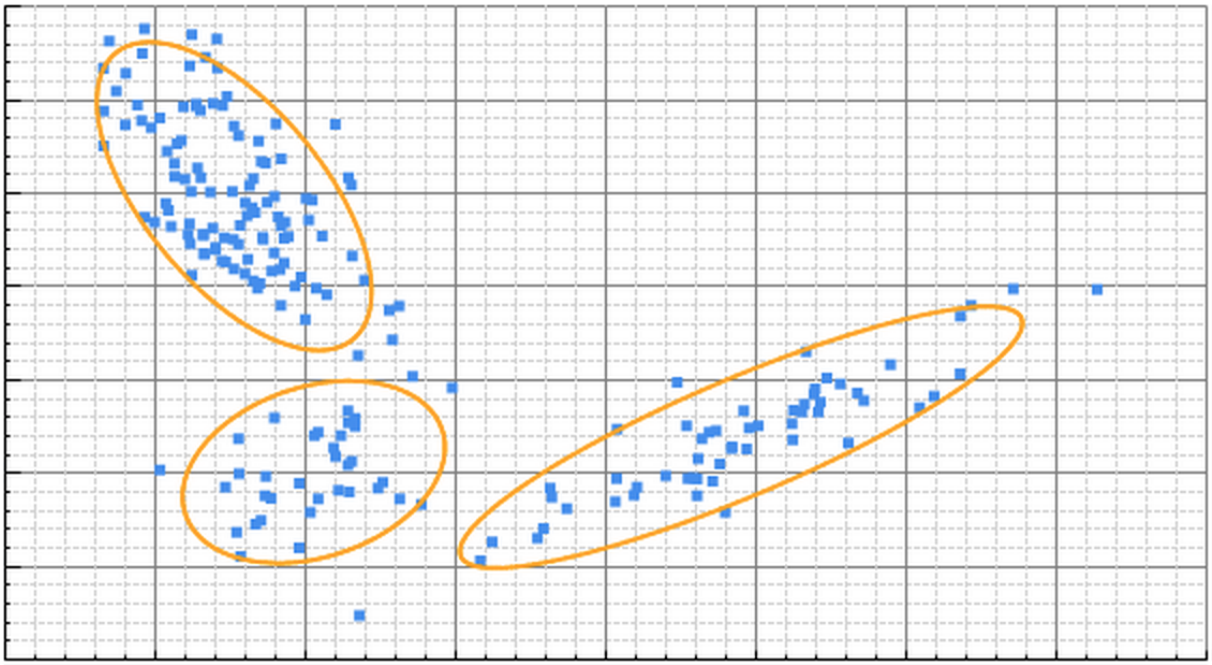

Figure 6:

We are also using facts from these knowledge bases to filter knowingly false statements to rank document that are using such statements with low rank. Our knowledge base is updated only when all given statements are knowingly true and only one or two statements are not defined yet. This information is supported by later when other resources publish the same statements. It is like voting for some statement by other authors. So if authors think that information is correct, they can approve it in their documents and it will update knowledge base.

Conclusion

In this paper we researched models, methods and approaches that we are using in our search engine. We described architecture of a crawling system which can run in parallel independently. Our system also collects metadata information from Web pages which is used to find relevant pages quickly. We also created knowledge base that uses popular open knowledge bases and it can also automatically populate with new knowledge.

Bibliography

[Gauch, 2003] Susan Gauch, Jason Chaffee, and Alaxander Pretschner. 2003. Ontology-based personalized search and browsing. Web Intelli. and Agent Sys. 1, 3-4 (December 2003), 219-234.

[Miner, 2012] Donald Miner and Adam Shook. 2012. Mapreduce Design Patterns: Building Effective Algorithms and Analytics for Hadoop and Other Systems (1st ed.). O'Reilly Media, Inc..

[Manning, 2008] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. 2008. Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA.

[Takhirov, 2012] Naimdjon Takhirov, Fabien Duchateau, and Trond Aalberg. 2012. An evidence-based verification approach to extract entities and relations for knowledge base population. In Proceedings of the 11th international conference on The Semantic Web - Volume Part I (ISWC'12), Philippe Cudre-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, and Jerome Euzenat (Eds.), Vol. Part I. Springer-Verlag, Berlin, Heidelberg, 575-590

Authors' Information

Oleksandr Kuzomin *- Prof. Dr., Department of Informatics, Office of Innovations & Marketing, Kharkiv National University of Radio Electronics; Kharkiv, Ukraine; tel.: +38(057)7021515; e-mail: oleksandr.kuzomin@gmail.com*

Major Fields of Scientific Research: General theoretical information research, Decision Making, Emergency Prevention, Data Mining, Business Informatics

Bohdan Tkachenko *- PhD degree student, Kharkiv National University of Radio Electronics; Kharkiv; Ukraine; e-mail: bohdan@tkachenko.io*

Major Fields of Scientific Research: General theoretical information research, Knowledge Discovery and Engineering, Data Mining, Business Informatics