CS297 Report
Article Generation using the Web

Gaurang Patel
gaurangtpatel@gmail.com

Advisor: Dr. Chris Pollett
Department of Computer Science
San Jose State University
Spring 2009

Table of Contents

Introduction

The web is a huge source of information, but the contents on the web are not organized. Search engines can search for useful information and present it in the form of a list of web links to the user. But search engines do not organize information from different websites into a coherent resource like a book. An article generation application is an intelligent mining engine that looks for the web content, combines and organizes the information in a meaningful way to generate an article. For CS297/CS298 project, we will make such an article generation tool. It will provide free articles to people based on their requirements. It generates article on the topic entered by the user using information available on the web. Information retrieval, semantic web and information extraction approaches will be used to develop the application. Generated articles will be in an electronic format i.e., e-book or web material/tutorial in form of web content. The articles will have well defined sections. Each of the section will talk about different aspect of the topic. Different sections can be technically thought as different clusters found while searching the material on web. Sections can have subsections and subsections can have sub subsections and so on till the desired depth. Details of the topic and desired depth of information to be covered may be the input parameters of the system.

Remaining part of the report talks about my approach working on the project, three deliverables including their goals and implementations, conclusion and future work:

There are two main modules of the system: 1) Information Retrieval, 2) Clustering and Summarizing Engine. In CS297, I planned to have the basic building blocks of the system ready, which will be integrated together in CS298 work to have the Article generation system. The project started with reading of some articles and looking for tools and technologies which can be helpful in developing the system. Google API and Nutch suited best for Information Retrieval module. Nutch was chosen finally since it is open source and to avoid any dependencies on the internal working of the Information Retrieval engine. Nutch may be configured to satisfy any changes required by the article generation system. Then I started looking for clustering/summarizing tools and their APIs. We chose to use and Carrot2 for high level clustering purpose and Open Text summarizer for summarizing documents contents.

Deliverable1

Download Nutch web crawler and test various sample crawl scenarios.

Nutch is open source web-search software. It builds on Lucene Java, adding web-specifics, such as a crawler, a link-graph database, parsers for HTML and other document formats, etc. Aim for

this deliverable is to achieve web crawling using Nutch and store the crawled results which will be further used by Clustering engine.

Crawling

Nutch Configuration:

- Set agent name in /conf/nutch-default.xml file.

  ```
  http.agent.name = 'MY SPIDER NAME'
  ```

- Create a directory with a flat file of root urls. For example, to crawl the nutch site you might start with a file named urls/nutch containing the url of just the Nutch home page. All other Nutch pages should be reachable from this page. The urls/nutch file would thus contain:

  ```
  http://en.wikipedia.org/wiki/India
  ```

  This is the url from which nutch will start crawling.

- Set the domain to crawl for in conf/crawl-urlfilter.txt file

  ```
  # accept hosts in MY.DOMAIN.NAME
  +^http://([a-z0-9]*\.)*en.wikipedia.org/
  ```

- Create a crawl directory on the local system. Nutch crawled results will be stored in this directory.

Nutch supports command lines for crawling:

```
$ bin/nutch crawl urls -dir crawl -depth 3 -topN 50
```

Search the crawled results.

- Enable clustering plug-in in nutch-site.xml by adding following property:

  ```
  <property>
  <name>extension.clustering.carrot2.defaultLanguage</name>
   <value>en</value>
  <description>Two-letter ISO code of the language.
  http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt</descript
  ion>
   </property>
  ```

- Deploy the web application that comes with nutch to tomcat server and run it in browser.

In this sample run, http://www.yahoo.com was crawled till 5 levels starting from url "http://sports.yahoo.com", fetching top 1000 results at each level.

Command to crawl:

```
$ bin/nutch crawl urls -dir crawl.sports.yahoo-5-1000 -depth 5 –topN
1000
```

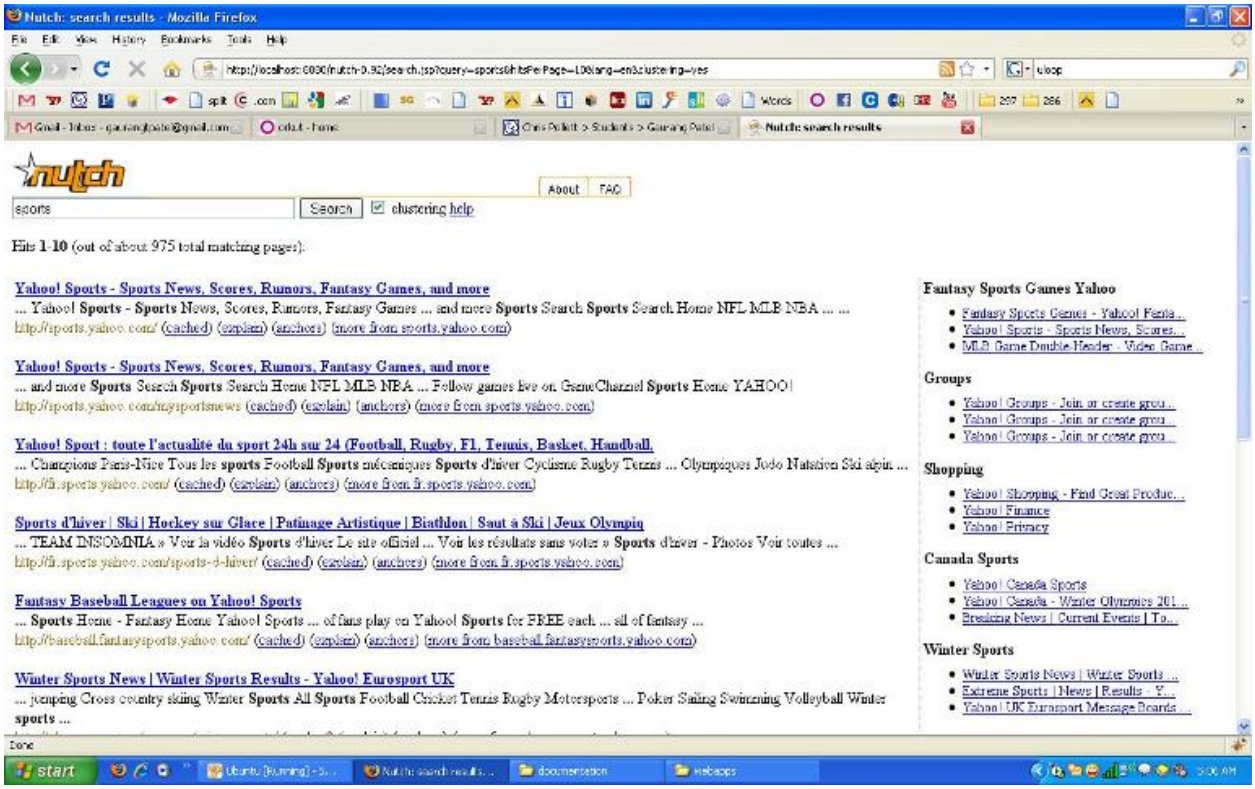


Figure 1: Search and Clustering results using Nutch and Carrot2

Left panel on the page in figure 1 shows the search results for query “sports”. Groups on the right panel of the page are the clusters found in these search results, if "clustering help" check box is selected. It uses Carrot2 clustering plug-in that comes with nutch.

Deliverable2

Download Open text summarizer, build it and achieve the summarizing goal.

Automatic text summarization is the technique, where a computer program summarizes a document. Summarizing of text and getting important contents from a bunch of sentences is an

important module for the article generation engine. Goal of this deliverable was to have demo of a tool which can be used to summarize paragraphs of texts.

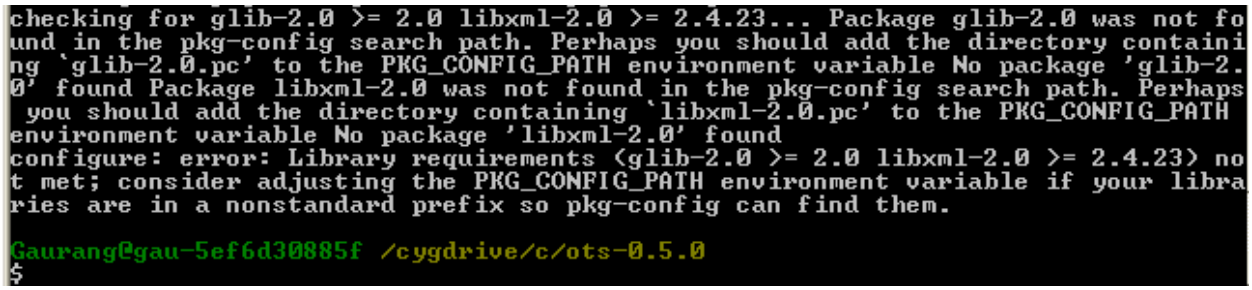
The Open Text Summarizer (http://libots.sourceforge.net/) is an open source tool for summarizing texts. The program reads a text and decides which sentences are important and which are not. Ots-0.5.0 is the actual version being used in this project.

Building ots-0.5.0

– ./configure

While running the configure script, I got some configurations errors as follows:

- glib and libxml library error



It is some versioning issue with xmllib and glib libraries. It seems that ots does not works with the latest version of libxml library. I searched for older versions of the library on internet and installed one of them, but it could not remove the library error.

Solution: Finally I skipped library version checks in configure script to bypass this check.

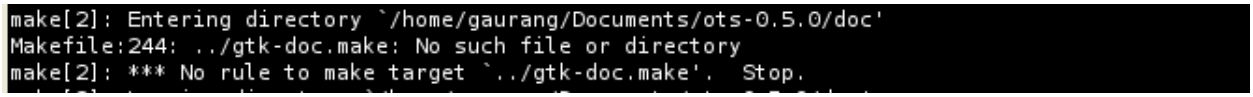
- popt library error

Solution: Installed libpopt0 1.14.0, and it worked fine.

– `$ make`

Make showed up with following errors:

- gtk-doc.make file error



This error was concerning requirement of gtk-doc.make file and its library.

Solution: Downloaded doc-gtk.make file and copied to ots root dir.
Downloaded doc-gtk-tools library. Configured, made and installed

- Error[1]: *** Invalid separator in doc-gtk.make file.

This error is concerning about the format of the file on different operating systems.

Solution: `$ unexpand doc-gtk.make`
It basically converts any tabs in the file to spaces.

– `$ sudo make install`

It created executable in usr/local/lib/ots

– `$ export LD_LIBRARY_PATH=/usr/local/lib`

To set the path.

OTS sample run

```
$ ots articles/sacbee1.txt--html
```

This will summarize the sacbee1.text file and generates the summarized text output in html format. Highlighted text in yellow color in figure 2 shows the summarized text from the text file.

Output:

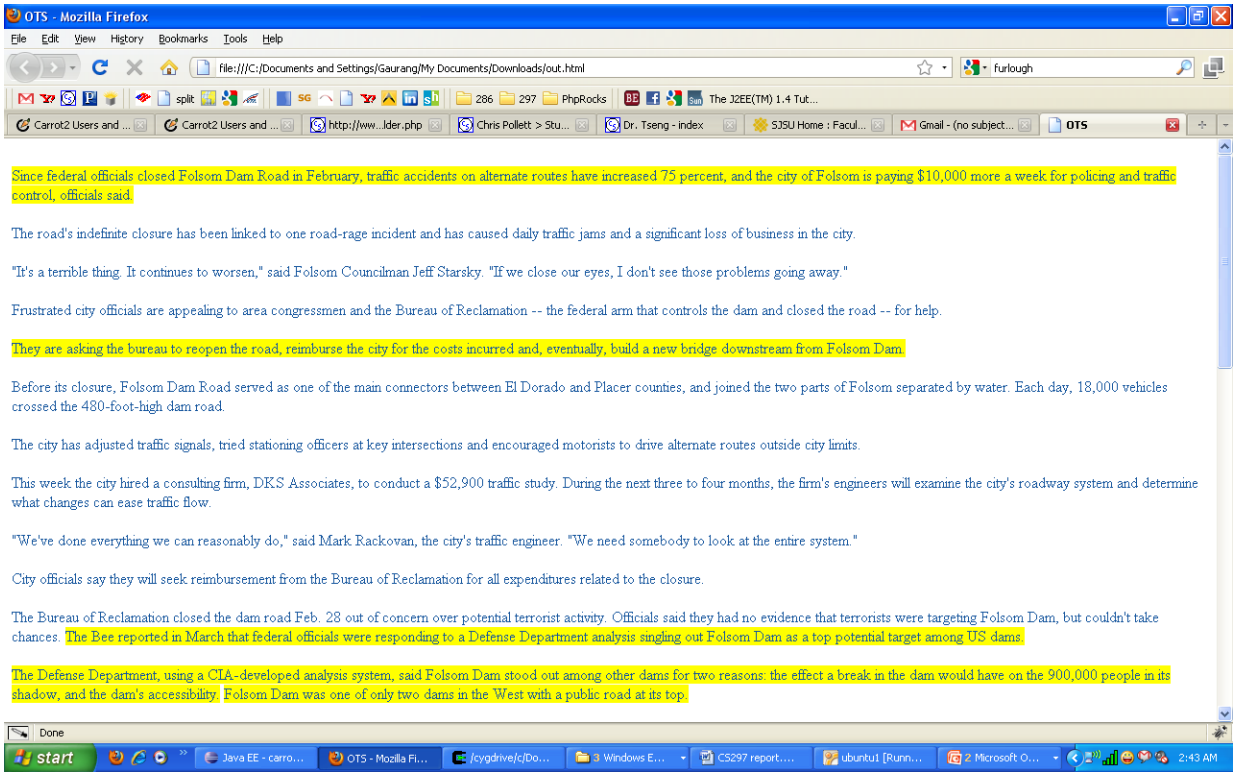


Figure 2: Sample ots output

Deliverable3

Make the carrot2 clustering work for document contents clustering.

Nutch returns search results which is basically a list of web pages. These pages need to be looked in detail for their contents for generating article out of them. Organizing search results into proper groups is the first step towards generating article. In this way, we are dividing the information in very high level groups. Carrot^2 is an Open Source Search Results Clustering Engine. It can automatically organize small collections of documents, e.g. search results, into thematic categories. Clustering can play important role in article generation. Nutch web crawler comes with Carrot2 plug-in to cluster the results returns by nutch. In this deliverable, we are looking for the possibility of using nutch for document level clustering. Document level clustering is basically to cluster the contents of a web page to organize the information on that page. Aim of this deliverable is to hack the carrto2 code to make it work for document level clustering. List of documents to be clustered is one of the input parameters to carrot clustering engine. I developed an API that breaks a web page into sub documents. Output of this API can be passed as input to Carrot.

I started looking into carrot code to find the appropriate place for new API into system to put to.

I observed following flow of code in the carrot2 codebase as illustrated in figure 3.

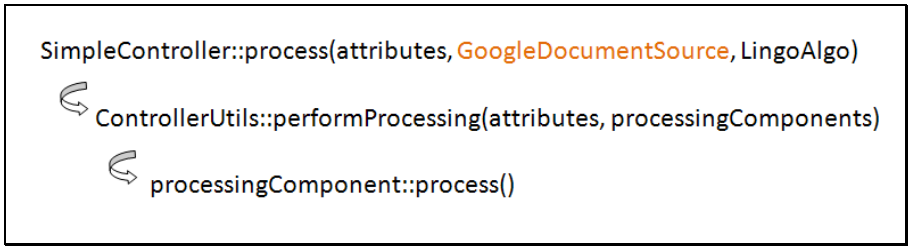


Figure 3: Flow of clustering code in Carrot2

SampleController, which is entry point of clustering, receives GoogleDocumentSource.class as an argument. GoogleDocumentSource.class is a java file in a carrot2 sub project carrot2-source-google. This class is responsible for getting search results from Google and organizing them into list of documents that can be understood by Carrot. Carrot has support for several search engines. It has separate sub projects for all search engines it supports. E.g. carrot2-source-google, carrot2-source-microsoft, carrot2-source-lucene, etc. So there is a need for a new API which can divide a document into sub documents and generate a list of documents understandable by carrot. I decided to have carrot2-source-document API and started working on implementation.

I created a new file ClusteringDocument.java in carrot2-examples project. This example can be run to demonstrate the document content level clustering using carrot.

Added a new method `getDocumentsFromFile(String pageURL)`. This method divides the inputted page into sub documents and returns the list of sub documents understandable by carrot clustering algorithm. Figure 4 show a snippet from the code of this method.

```
        int pageLength = pageString.length();
        int docsCnt = pageLength/500;
        String [][] docContent = new String [docsCnt][3];

        int i =0;
        for(i=0;i<docsCnt;i++) {
                String docText = pageString.substring(i*500, (i+1)*500 -1 < pageLength ? (i+1)*500 -1 :
pageLength-1);
                docContent[i][0] = docText.substring(0,50);
                docContent[i][1] = docText;
                docContent[i][2] = "";
        }

         List<Document> documents = new ArrayList<Document>();
         for (final String [] element : docContent)
             documents.add(new Document(element[0], element[1], element[2]));

        return Collections.unmodifiableList(documents);
```

Figure 4: Code snippet of getDocumentFromFile() mehtod

Clustering output for url: http://en.wikipedia.org/wiki/Data_mining

Figure 5 shows clustering results.

```
Attributes:
processing-time-total:   125
processing-time-algorithm:   125
2009-08-26 15:21:00,937 INFO  org.carrot2.clustering.lingo.LingoClusteringAlgorithm: Native BLAS routines not available
Collected 79 documents


Created 27 clusters

Geographic Data (7 documents)
  [45]  So far, data mining and Geographic Information Sy

  [48] ts, that are conventionally archived in hybrid dat

  [49] de ill-structured data such as imagery and geo-ref

  [50] rability, including differences in semantics, refe

  [51] eraction through attributed geographic space such

  [66] Ã©n GN, Bate A, Hopstadius J, Star K, Edwards IR.

  [67] ., (eds.), 1999, Spatial Multimedia and Virtual Re

International Conference (6 documents)
  [12] Information Technology and Decision Making summari

  [13] her Computer Science conferences on data mining in

  [59]  for the Field of Data Mining and Knowledge Discov
```

Figure 5: Clustering results for webpage: http://en.wikipedia.org/wiki/Data_mining

Challenges during the deliverable:

Carrot2 is a quite big project. I checked out stable branch of the project, and I got a total of 65 sub-projects/plug-ins and 700 java files. Hacking carrot wasn't easy, but Eclipse made it easy somehow.

Conclusion:

I created basic modules for the article generation system in CS297 coursework. Deliverable1 achieved the goal of building the crawler for the system. This module will be needed in accomplishing deliverable 1 for CS298. Deliverable 2 and 3 are the building blocks of the Clustering Engine module of the system and they will play an important role in achieving deliverable 1 for CS298. Moreover, working on these sub modules enhanced my knowledge of several technical areas, tools and IDEs.

CS298:

Individual modules of web crawler and Clustering were developed during CS297. Integration of these modules (Deliverable1 for CS298) to have the final article generation system will be the primary goal for CS298 coursework. Another goal (Deliverable 2 for CS298) would be to refine the clustering sub module. This will require optimizing the clustering and summarizing engines in order to produce better organized and content efficient articles. This might require hacking code of Carrot2 and/or OTS, or even scoping for a new tool.

References:

[2007] Web Page Analysis: Experiments Based on Discussion and Purchase Web Patterns. Jana Kocibova, Karel Klos, Ondrej Lehecka, Milos Kudelka, Vaclav Snasel. 2007 IEEE/WIC/ACM International Conferences on Web Intelligence

[2007] A Method for Integration of Web Applications Based on Information Extraction. Hao Han and Takehiro Tokuda. Eighth International Conference on Web Engineering.

[2007] A Novel Method for Hierarchical Clustering of Search Results. Gang Zhang Yue Liu Songbo Tan Xueqi Cheng. 2007 IEEE/WIC/ACM International Conferences on Web Intelligence

[2008] Open Text Summarizer http://libots.sourceforge.net/

[2009] Nutch Wiki http://wiki.apache.org/nutch/NutchTutorial

[2009] Nutch http://lucene.apache.org/nutch/

[2009] Carrot^2 Clustering Engine http://search.carrot2.org/stable/search