

Development of Specialized Search Engine

G22. 2580

Web Search Engines

Professor Ernest Davis

Team Members: Changle Wang, Yifei Sun

Objective

We want to develop a specialized web search engine with a specific topic, such as finance, NYU. Using a specialized search engine, users can get search results with better precision and recall. To build that kind of search engine, we need to firstly design a topic-oriented crawler, which could determine whether a page is relevant to the topic or not with the help of classifier. Then we need to index web pages downloaded by the crawler. At last, we implement a web interface to provide users with access to perform search.

Architecture

This system consists of three parts, topic-oriented crawler, indexer, and web interface. First we provide crawler with a seed file to let crawler begin to work. The crawler continuously downloads web pages from Internet, then use classifier to determine if pages are relevant to topic. If yes, crawler then extract all out links from that page, and add them to link pool; otherwise, just discard them. After crawler complete its job, we use html parser to extract text content from html documents for indexer to index. At last, we develop a web interface for users so that they can perform query in a user-friendly way. Also we create an interface for system administrator. Administrator could initiate index, create index, and delete index online. The following figure indicates how our system works.

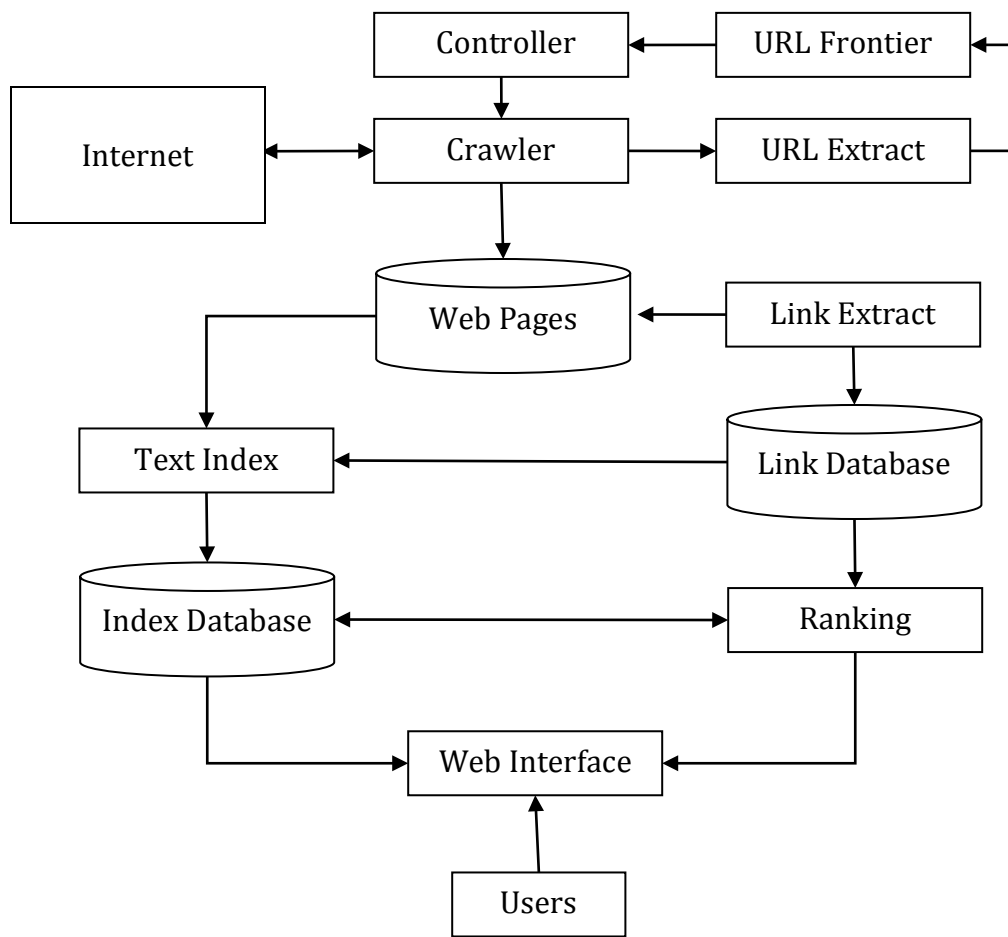


Figure 1 Search Engine Architecture

Classifier

In order to obtain the topic-oriented crawler, at first we need to implement a classifier, which could determined whether a text file is relevant to the topic or not. There are several popular classification algorithms. In this experiment, we test three different algorithms, Support Vector Machine, KNN, Naive Bayes Algorithm.

The first step is to select the corpus for training and testing. In this experiment, we set “finance” as the topic. We select 20 Newsgroups and Reuters-21578 dataset as training and testing corpus. 20 Newsgroups corpus consists of 20 different topic news, including computer, music and sports. Reuters corpus contain a lot of economics related news. So we use 20 Newsgroups as negative

corpus, and Reuters corpus as positive corpus. We randomly select 50 articles from 20 Newsgroups for each topic, and select 1000 articles from Reuters to constitute the training corpus. Similarly, we select 1355 article from this two corpus to constitute the test corpus.

Pre-process

First, we tokenized all the news, and delete all numbers, symbols. Then we delete all tokens whose length is less than 3. We also filter out the word in stop word list (<http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop>). At last, we apply stemmer algorithm (using SnowballAnalyzer) to the remaining words. Now we get tokens we need. Then we select top 2000 tokens, which appear most time in the corpus. At that time, articles all become 2000-dimension vectors, whose dimension corresponds to a different term. Then we use tf-idf weighting method to assign weight to each term. In order to avoid article's length affecting weight, and avoid some term dominating the vector, we use following formula to normalize each vector.

$$W_{ik} = \frac{\sqrt{tf_{ik} \times \log(\frac{N}{n_k} + 0.01)}}{\sqrt{\sum_{k=1}^n (tf_{ik}) \times \log(\frac{N}{n_k} + 0.01)}}$$

W_{ik} is the term k 's weight in document i . tf_{ik} means the number of times the term k occurs in the document i . N is the total number of corpus. n_k is the number of documents which contains term k in the corpus. Finally, we get the training and test dataset we need.

Classification

As we said before, we test three classification algorithms, Support Vector Machine, KNN, Naïve Bayes Algorithm. Each algorithm has its unique theory support. We use libsvm library to perform the support vector machine

classification, use java-ml package to do the KNN and Naïve Bayes Classification. The following is the classification result comparison.

Algorithms	Correct	Wrong	Accuracy
Naive Bayes	758	597	0.5594096
KNN(5)	1347	8	0.9940959
LibSVM	1349	6	0.995572

From the result, we can see KNN and LibSVM classification perform a lot better than Naïve Bayes. Although KNN achieve close accuracy with LibSVM, it runs far slower than LibSVM. So we decide to use LibSVM algorithm in our project.

Topic-Oriented Crawler

Topic-Oriented Crawler plays an important role in a specialized search engine. It can collect relevant documents and filter out irrelevant documents based on a classifier. First it starts to work from a seed list we provide. The crawler downloads the web pages it can reach, then extracts text content from web pages, convert text content to vector model, and perform libsvm classification to that vector. If it's relevant to topic, the crawler extract all the out links in that page, and add them into frontier. Figure 2 describes how this crawler works.

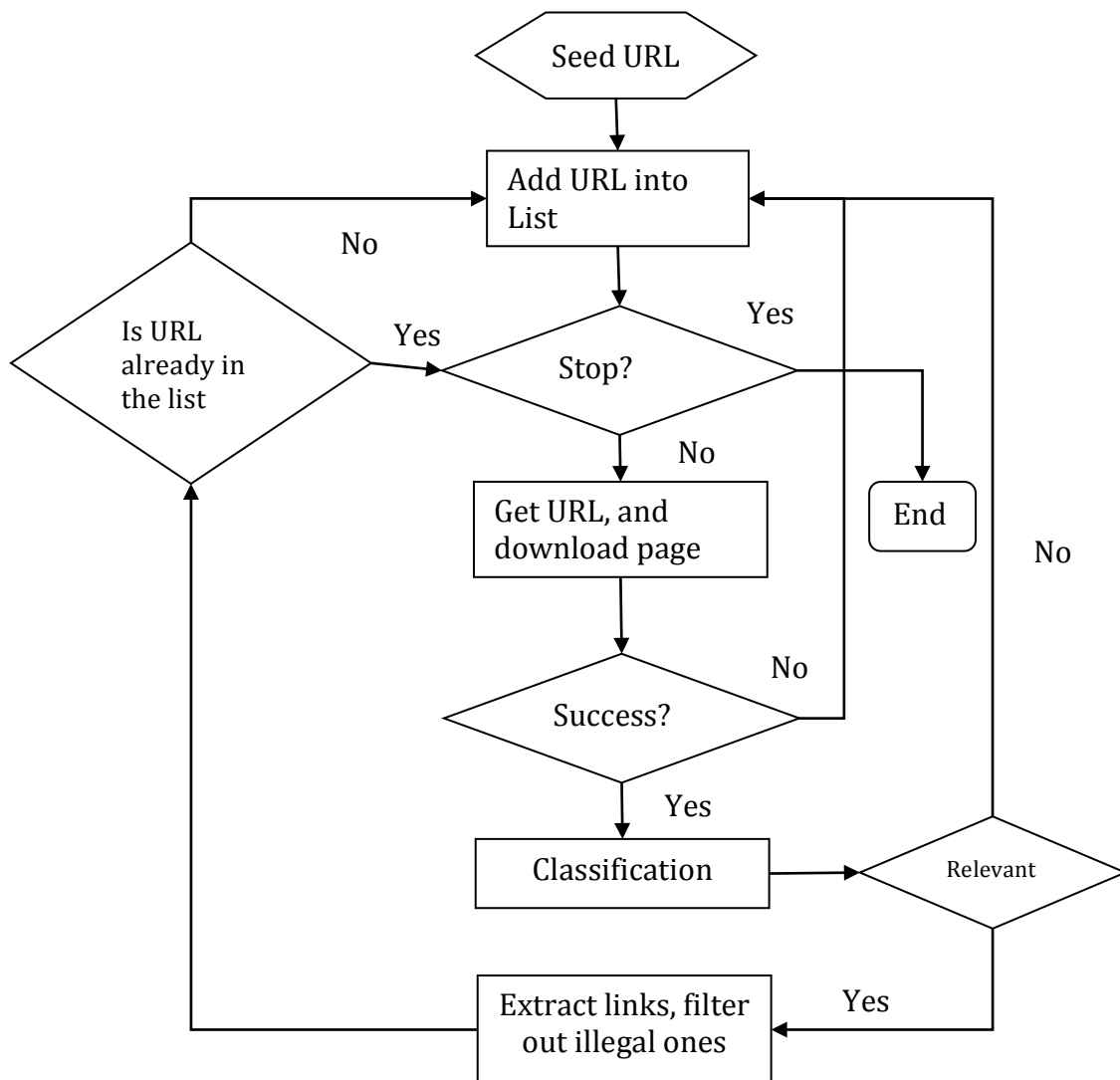
In this project, we use an open source crawler—Heritrix. But, Heritrix is not a topic-oriented crawler, so we need to add classification module to it. In order to let the crawler has the ability to determine pages' relevance, we added and modified some source code in Heritrix. The key part is that we add a new extractor into it. So when it downloads a web page, we run that extractor. Firstly, it will get text content from that page, then it will perform classification function, then works as the steps we describe previously.

Key code:

```

String vector;
int classType=-1;
vector = Classifier.convertToVector(content,stopDir, featureDir,
rangeDir);
classType=Classifier.predict(vector, modelDir);
if(classType==1){
    System.out.println("This page seems to be relevant,
extract out links");
    extract(curi, cs);
    curi.linkExtractorFinished();
}else{ System.out.println("This page seems to be irrelevant,
discard");}

```



Build Index

After we get all web pages we need, we need to build index for them, so that users could perform query more efficiently and effectively. In this project, we use an open source full-feature text search engine library—Apache Lucene to help us build the inverted index.

We create several fields for each indexed file, including id, link, title, and contents. Those fields are useful when users want to make queries on them. To build index, we firstly extract text content from each html file, generate link, then use lucene document to represent them.

Lucene's index process begins from IndexWriter. Every time it reach a new document, IndexWriter create a new Document, add fields into document. After doing this, IndexWriter use addDocument() method to add document into index. At last, IndexWriter call optimizes function to do optimization.

Figure 3 indicates the index process of lucene IndexWriter.

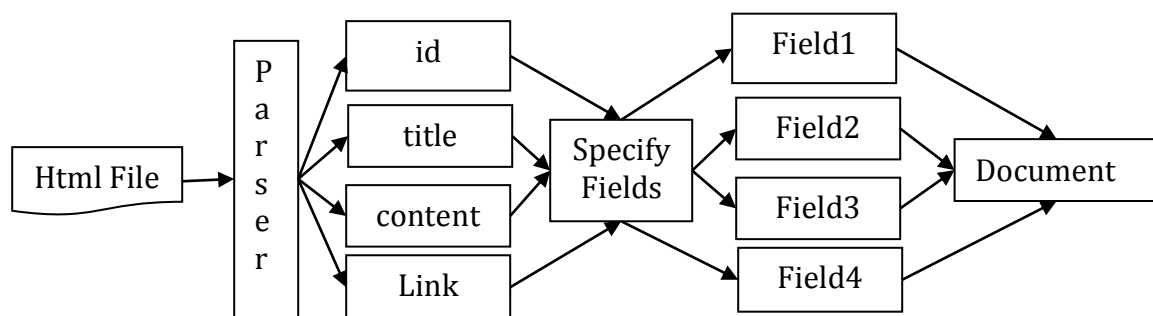


Figure 3 Index Process

Build Searcher

Search Module is the interface between users and index. The search module could receive users' queries, tokenize queries into separate terms, perform different search scheme on inverted index. First of all, Query from users will be sent to analyzer. After analyzer's process, queries become tokens. Then we perform search function to look up to documents' different fields. At last, we return users the ranked result list.

Key code:

```
IndexSearcher searcher=new IndexSearcher(indexPath); //Create
Searcher
//Create QueryParser object, and initialize.
QueryParser parser=new QueryParser(field,new StandardAnalyzer());
//Use QueryParser to parse users' query
Query q=parser.parse(searchkeys);
//Generate Hits collection according to specific query
h=searcher.search(q);
```

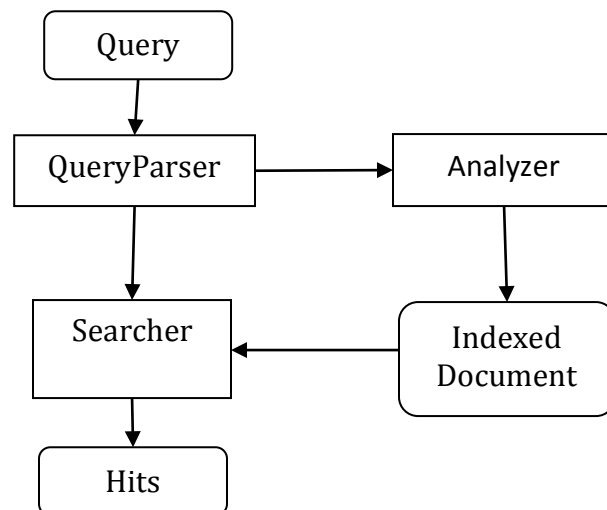


Figure 4 Search Process

Categorization

WanderLust also offers tools, which can categorize a piece of financial news into 5 categories (bonds, stocks, commodities, options and currencies).

In order to get training text set, wrote a crawler for Yahoo! Finance, and wrote some specified HTML parsers of Reuters, WallStreetJournal, MarketWatch and so on. As a result, we obtain about 1,000 articles of each category for training.

Regarding the text training, we have chosen the Rocchio Classification algorithm. At the beginning, we have 5 groups of user-labeled training text. (We have automatically labeled them via crawling.) Then we parse the documents by SnowBallAnalyzer, and calculate the word frequencies in total, and pick the first 2,000 terms to be the features. Next step, for each article, we use the tf-idf weighting method to assign weight to each term appeared and generate corresponding vector of every article. At last, we calculate the centroid of a group, and get 5 centroid vectors eventually.

When a new article comes, first we convert it into a vector by tf-idf weighting algorithm and compare it with the 5 centroid-vectors, choose the “nearest” centroid’s category as the new article’s category.

In our project, the crawler is banned by some famous financial news website, so it is difficult to integrate this module with the main search engine, because most of the NYU’s news are financial non-related documents. So we just put it as a regular button on the index page. In future, if we get rid of the crawler problem, we want to change this algorithm into K-means clustering algorithm (Use the 5 centroid-vectors as the initial means and let the other vectors converge automatically) and other algorithm to implement clustering.

Sample Search Result

In this project, we modified a crawler, making it has the potential to only download web pages relevant to the topic we set. However, because our training corpus is old, and most financial news website don't allow common crawler to download news pages, so we let crawler only download web pages from NYU Computer Science Department and NYU.edu. As the result, all queries search through index file we build from web pages we downloaded. So your query results would all related to NYU.



Figure 5 Web Index Page

We also support advanced search. You can specify your query type such as accurate, fuzzy, and prefix mode. Also you can set the result number of per page, and so on.




Type
 Search results number per page
 Search Result Setting
 Search File Type

Figure 6 Advanced Search Page

The following is query result of search “Computer” keyword.

Wanderluster Web Search Engine



computer

About 370 Results

[New York University - Women in Computing](#) <WebHtml>
 New York University - Women in **Computing** This Web page uses frames. Your current Web browser does not display frames or frame viewing has been turned off.
<http://cs.nyu.edu/~wincweb1/winc.html>

[Computer Science Colloquium - NYU Computer Science Department](#) <WebHtml>
Computer Science Colloquium - NYU **Computer** Science Department Search Location Contacts Directions NYC Information Admissions Undergraduate Admissions Graduate Admissions People Faculty Researchers
<http://cs.nyu.edu/web/Calendar/colloquium/fall08/oct10.html>

[Computational Plasma Physics](#) <WebHtml>
Computational Plasma Physics Professor Paul R. Garabedian Director, Division of **Computational** Fluid Dynamics Courant Institute of Mathematical Sciences New York University 251 Mercer Street New York
<http://www.math.nyu.edu/faculty/garabedi/index.html>

[NYU Computer Science Department](#) <WebHtml>
 NYU **Computer** Science Department Search Location Contacts Directions NYC Information Admissions Undergraduate Admissions Graduate Admissions People Faculty Researchers/Visitors Administration/Staff
<http://cs.nyu.edu/csweb/People/amirpnueli.html>

[NYU Computer Science Department](#) <WebHtml>
 NYU **Computer** Science Department Search Location Contacts Directions NYC Information Admissions Undergraduate Admissions Graduate Admissions People Faculty Researchers/Visitors Administration/Staff
<http://cs.nyu.edu/web/People/amirpnueli.html>

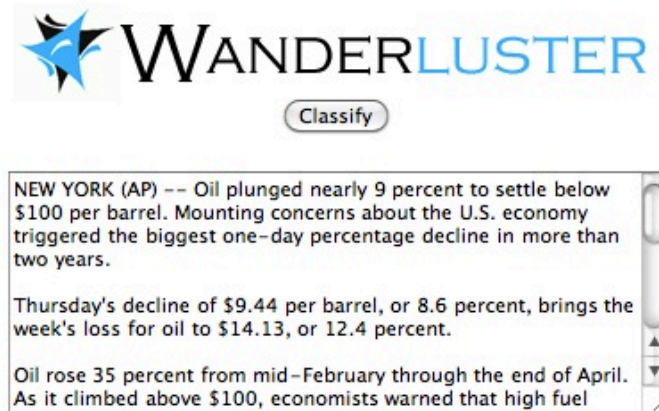
[Welcome - NYU Computer Science Department](#) <WebHtml>
 Welcome - NYU **Computer** Science Department Search Location Contacts Directions NYC Information Admissions Undergraduate Admissions Graduate Admissions People Faculty Researchers/Visitors
<http://cs.nyu.edu/csweb/index.html>

[Welcome - NYU Computer Science Department](#) <WebHtml>
 Welcome - NYU **Computer** Science Department Search Location Contacts Directions NYC Information Admissions Undergraduate Admissions Graduate Admissions People Faculty Researchers/Visitors
<http://cs.nyu.edu/web/index.html>

Figure 7 Sample Search Result

We also provide a web interface for classification. Our classifier is trained on 20Newsgroup and Reuters corpus. It's a binary classifier, which could determine whether a news is financial relevant or not.

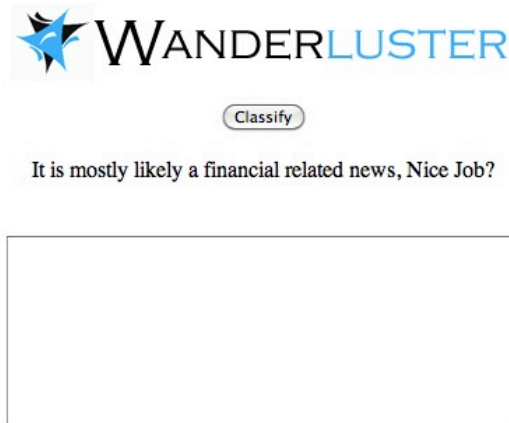
The following figure is that we classify a news which we copy from Yahoo! Finance website to show the classification result.



[Sample Positive Corpus](#) [Sample Negative Corpus](#)

The classifier is trained base on 20Newsgroup corpus(20 different topics) and Reuters 21578 corpus(Mainly economics related)

To better verify accuracy of this classifier, please select corpus from these copura



[Sample Positive Corpus](#) [Sample Negative Corpus](#)

The classifier is trained base on 20Newsgroup corpus(20 different topics) and Reuters 21578 corpus(Mainly economics related)

To better verify accuracy of this classifier, please select corpus from these copura

Figure 8 Classification Result

Figure 9 to Figure 11 show the results of categorization, it's a financial news about stock, which copied from MarketWatch.com.

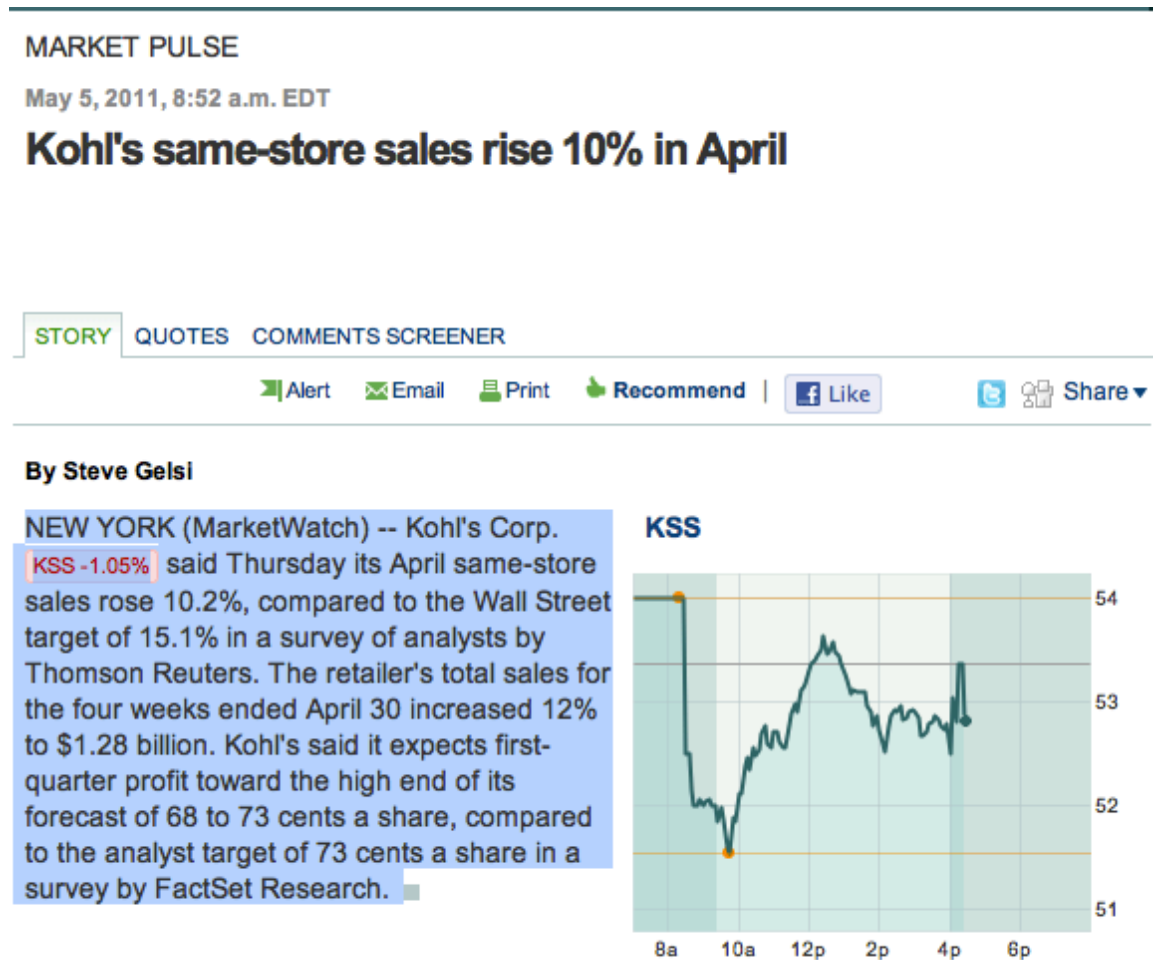


Figure 9 This is a news copied from MarketWatch.com



This tool can categorize a financial article into five categories about U.S. Markets (Stocks, Bonds, Commodities, Currencies, Options). Copy a piece of financial news into the box below and click to cluster it.

Check it out!

Categorize it!

NEW YORK (MarketWatch) -- Kohl's Corp. KSS -1.05% said Thursday its April same-store sales rose 10.2%, compared to the Wall Street target of 15.1% in a survey of analysts by Thomson Reuters. The retailer's total sales for the four weeks ended April 30 increased 12% to \$1.28 billion. Kohl's said it expects first-quarter profit toward the high end of its forecast of 68 to 73 cents a share, compared to the analyst target of 73 cents a share in a survey by FactSet Research.

This tool is based on [Rocchio Classification](#) algorithm, the training texts come from [Yahoo! Finance](#). They are: [Stocks](#), [Bonds](#), [Commodities](#), [Currencies](#), [Options](#)

Figure 10 Copy the news into the text field



Categorize it

Category: stocks

A large, empty rectangular text box with a thin black border and a small cursor icon in the bottom right corner.

This tool is based on [Rocchio Classification](#) algorithm, the training texts come from [Yahoo! Finance](#). They are: [Stocks](#), [Bonds](#), [Commodities](#), [Currencies](#), [Options](#)

Figure 11 It's categorized correctly!

Future Work

Our classifier is trained on 20Newsgroup and Reuters corpus. Although it performs well on these test dataset, it performs not that well when it turn to current news. I guess it's because that the news format change along the time, since the training data are all old news, so add recent news could increase the classification accuracy.

In this project, we only index files with extension such like "html,php,asp,jsp", but a lot of files we download don't have that kind of extension. So we could improve index's algorithm to make better use of all documents we download.

Lucene is a text search engine, so its ranking algorithm may not reflect the importance of web pages. In order to improve ranking, we could modify ranking algorithm by adding PageRank module.

Because of time limitation, we don't add clustering into result pages. In future, we could apply clustering to query result, provide users with better result representation.

Development Environment

Programming Language: Java

Web Server: Apache Tomcat

IDE: Eclipse

Text Search Engine Library: Apache Lucene

Crawler: Heritrix

Html Extract Tool: HtmlParser

Analyzer: SnowballAnalyzer

Web Structure: Struts2.0, Spring