New York Times Newspaper Archive

CS6913 Web Search Engines Course Project

Spring 2013

Jiankai Dang 0482923
Binbin Lin 0490388

Table of Contents

1. Introduction

The NYT allows academic researchers to work with a collection of all articles published between 1987 and 2007, more than one million articles in total. We got this data set from our university, and then built this system that allows people to browse news using search, Google Maps, and timelines, according to the extracted geographic information and time stamps from the meta data of articles.

2. How to Implement a Search Engine: Build Index

An inverted index, also named posting files or inverted files, is an index data structure storing mapping from each terms to content include it. For example, words (or word id) to its location in a set of documents or in a database files.The purpose of inverted index is to fast full text searches. [1] For example, Document 1 is [“Inverted Index”], Document 2 is [“Inverted Index is a kind of Index”]. The inverted index of word “Index” is [[1, 1], [2, 2]], meaning that this word is in document 1 for 1 time and in document 2 in 2 times.

- **Build Index**

Suppose the memory is large enough, we could scan all files and add document id into inverted index of terms within the document. Unluckily, in practice memory is probably not enough. We would tell what we do in “efficient building”. We will introduce some useful or highlights details of our project.

- **Parsing the Collection**

When building index, we should make a decision whether build index for non-natural word, such as “0fxxx”. If not, we could use Natural Language Toolkit [5] to filter. This would make inverted index in a small size. The speed would be highly increased. If so, our user could get result of special search queries. For example, someone want to search a special line of code “string.rfind(x)”. However, the inverted index would be much larger. In this project, we choose a third way. We do build inverted index for every term. For efficient, we build up a cache for common words in natural language. This would be told later in details.

We will introduce three most important index as following:

a. Inverted Index

We use postingMap during building up the index. The postingMap is a tree map and it maps the word to another tree map, and the internal tree map contains the document ID to its term frequency in this document. The reason we use tree map is we can take the advantage of tree to output posting by order. We will output the postings first by the order of word after then by the order of document id for making chunks. We will output the tree when the node number reaches a given number, so the log(n) of search time and insert time wouldn't be a problem.

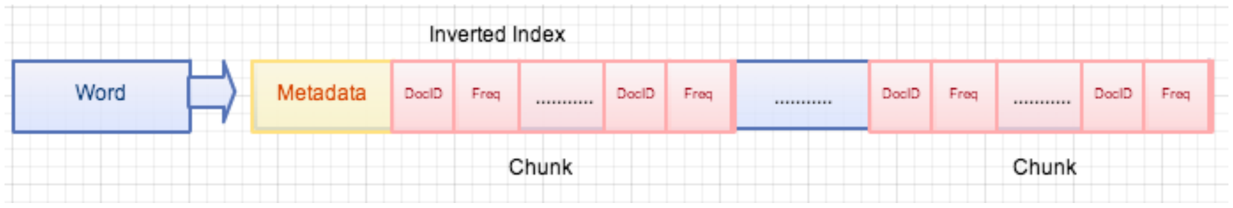


Figure 1

b. Lexicon Index

LexiconMap is a hash table mapping the word to an array containing the information of inverted list of the given word. The information includes document frequency of the given word, filename which stores this specific inverted list, the offset to the inverted list, the length of the inverted list and the doc ID's chunk number of the inverted list.

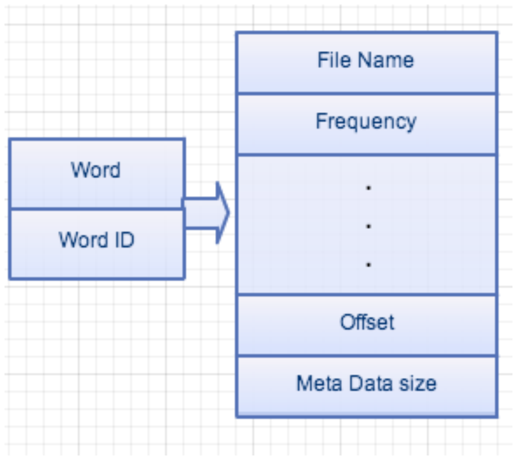


Figure 2

c. URL Index

urlDocMap is a hash table mapping a document id to an object of class UrlDocLen which consists of its url and the document text length. Since we need calculation the proportion of the document length to the average length of all documents. We record all the units in bytes.

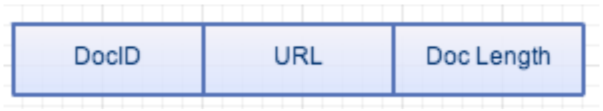


Figure 3

- **Efficient Building**

There are mainly four approaches to speed up index building: linked list, I/O efficient sort, merging indexes and lexicon partition. We use a mixed solution. Our solution is to divide files into several parts. We build up inverted index separately for each part. Finally, our program uses an I/O efficient algorithm to merge these inverted indexes (in most of time, the algorithm is merge sort). The disadvantage is speed is a little slower, for we build multiple semi-lexicon files. The advantage is scaling. Our program allow adding in new set of inverted index, we don't care where it is from.

Inverted files created by one crawler are available to merge into inverted files created by other crawlers (or maybe prior version of itself).

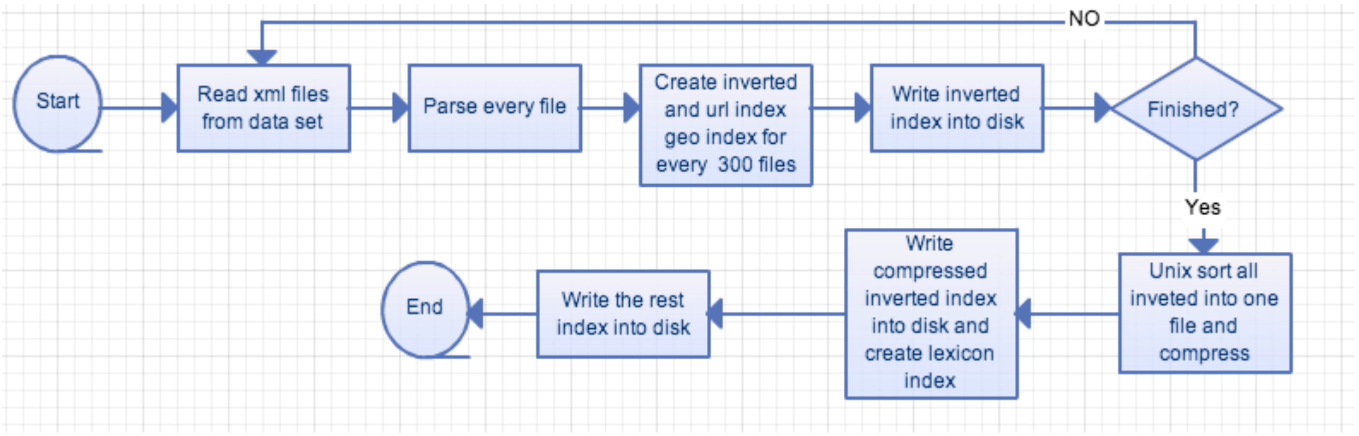

Start
Read xml files from data set
Parse every file
Create inverted and url index geo index for every 300 files
Write inverted index into disk
Finished?
NO
Yes
Unix sort all inveted into one file and compress
Write compressed inverted index into disk and create lexicon index
Write the rest index into disk
End

Figure 4 FlowChart of Index building

- **Index Compression**

The goal is to compress inverted index. More, we don't care the size of inverted index in hard drive, for the hard-drive is cheap. What we care is the size of cached inverted index. If the size of one inverted index is small, we could cache more. We also care about the speed of decompress.

There are many possible methods, e.g., VByte, Chunk-wise. Variable byte (VB) encoding uses an integral number of bytes to encode a gap. The last 7 bits of a byte are "play load" and encode part of the gap. The first bit of the byte is a continuation bit. It is set to 1 for last byte of the encoded gap and to 0 otherwise.

$$l = \lfloor \log_{128} n \rfloor$$

Where l is the number of bytes we would use and n is the number we want to compress.[2]

3. How to Implement a Search Engine: Query Processing

We assume that we have a document collection D = {d0, d1, …, dn - 1} of n web pages that have already been crawled and are available on disk. Let W = {w0, w1, …, wm – 1} be all the different words that occur anywhere in the collection. Typically, almost any text string that appears between separating symbols such as spaces, commas, etc., is treated as a valid word (or term) for indexing purposes in current engines.

Typically, almost any text string that appears between separating symbols such as spaces, commas, etc., is treated as a valid word (or term) for indexing purposes in current engines.

- **Query Types**

This project will answer two types of queries: One word queries: Queries that only contain one word, like soccer, or music. Free text queries: Queries that contain several words that are separated by space. This project uses AND query strategy, which returns a document if and only if all the terms appear in this document.

- **Query Parsing**

We notice that input query would not follow any rules. We could not treat it as a natural language format. For example, “Idon’tLikeyou” would exist as a query. Also, we should also expend some short-format word, like “it’s” should be “it” and “is”. Therefore, we use a NLP parser to parse every query. We also delete non-meaningful words like ‘*#$’.

- **Query Execution**

Given an inverted index, a query is executed by computing the scores of all documents in the intersection of the inverted lists for the query terms. This is most efficiently done in a document-at-a-time approach where we scan the inverted lists, and compute the scores of any document that is encountered in all lists. (This approach is more efficient than the term-at-a-time approach where we process the inverted lists one after the other.) Thus, scores are computed, and top-k scores are maintained in a heap structure. In the case of AND semantics, the cost of performing the arithmetic operations for computing scores is dominated by the cost of traversing the lists to find the documents in the intersection, since this intersection is usually much smaller than the complete lists.

- **Efficient Cache**

Our cache technique is based on natural language words frequency. We load an open English words frequency file (“frequency.txt) [4]. Depending on the word frequency, we cache inverted index of selected words with descending frequency.

- **Ranking function**

So far, our program could present the result set based on your query. However, there is still a big problem. We could not show hundreds of result pages to users. We should rank them and show top 10 or top 20 of them. Here we would discuss the ranking function we use: Okapi BM25.

Okapi BM25 is a ranking function used by search engines to rank matching documents according to their relevance to a given search query. It is based on the probabilistic retrieval framework developed in 1970s and 1980s by Stephen E.Robertson, Karen Sparck Jones and others [3]. The ranking function is:

$$BM25(q,d) = \sum_{t \epsilon q} \log(\frac{N - f_t + 0.5}{f_t + 0.5}) * \frac{(k_1 + 1) f_{d,t}}{K + f_{d,t}}$$

$$K = k_1 * ((1 - b) + b * \frac{|d|}{|d|_{avg}})$$

N: total number of documents in the collection;
f_t: number of documents that contain term t;
$f_{d,t}$: frequency of term t in document d;
$|d|$:length of document d;
$|d|_{avg}$: the average length of documents in the collection;
k_1 and b: constants, usually $k_1 = 1.2$ and $b = 0.75$.

- **Query Process**

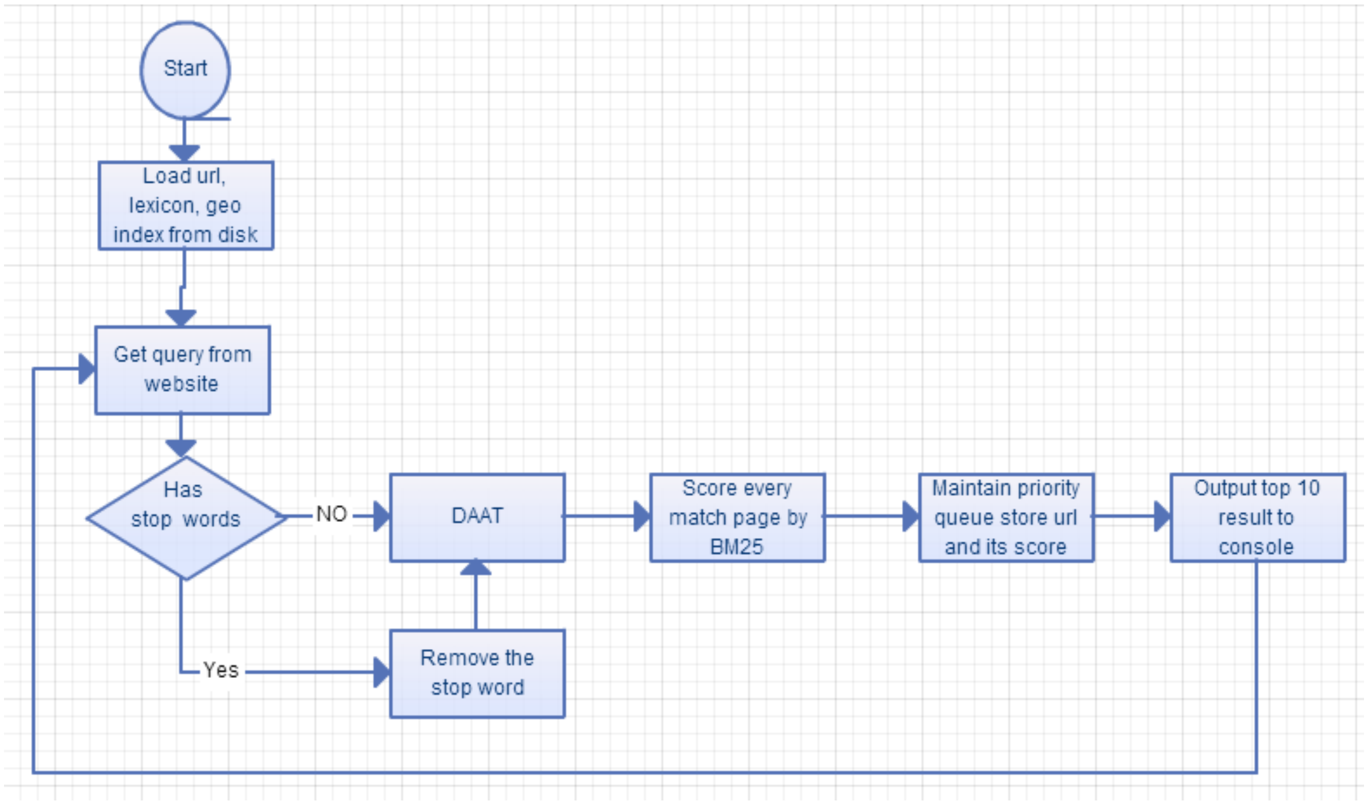

Start
Load url, lexicon, geo index from disk
Get query from website
Has stop words
NO
Yes
DAAT
Remove the stop word
Score every match page by BM25
Maintain priority queue store url and its score
Output top 10 result to console

Figure 5 Flow Chart of Query Process

4. About Programs: High-level Structure

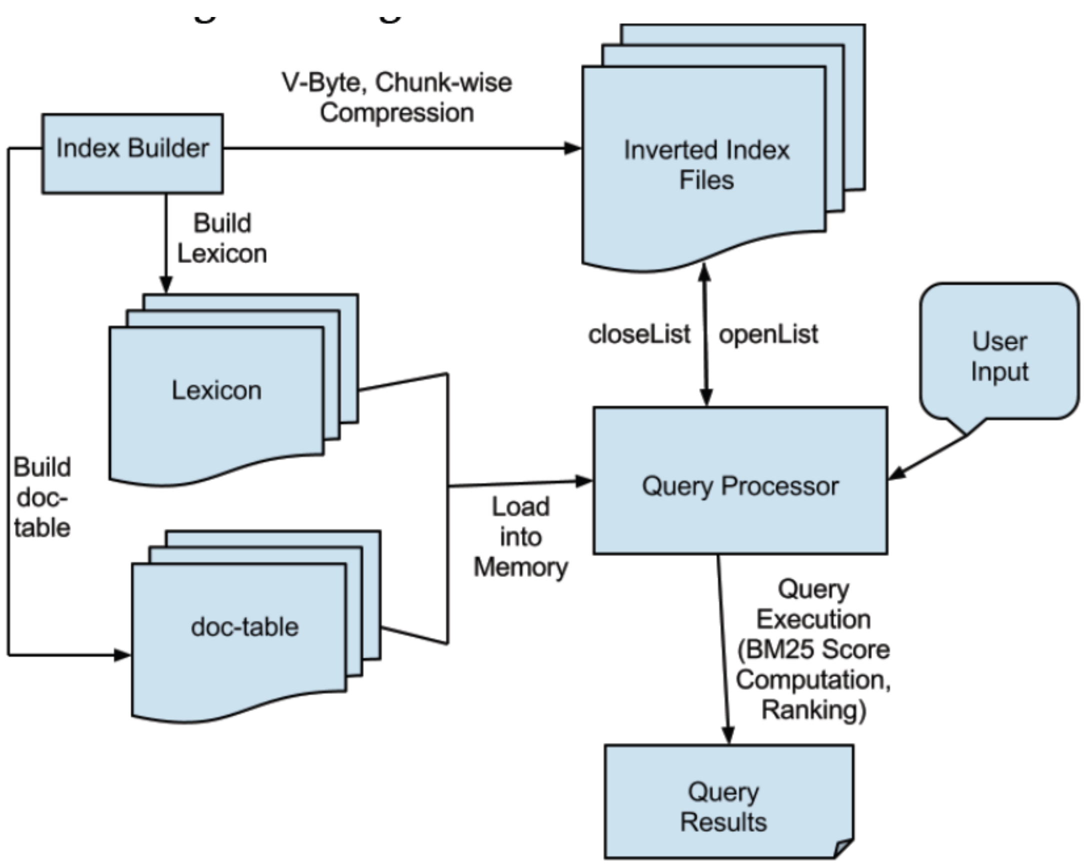


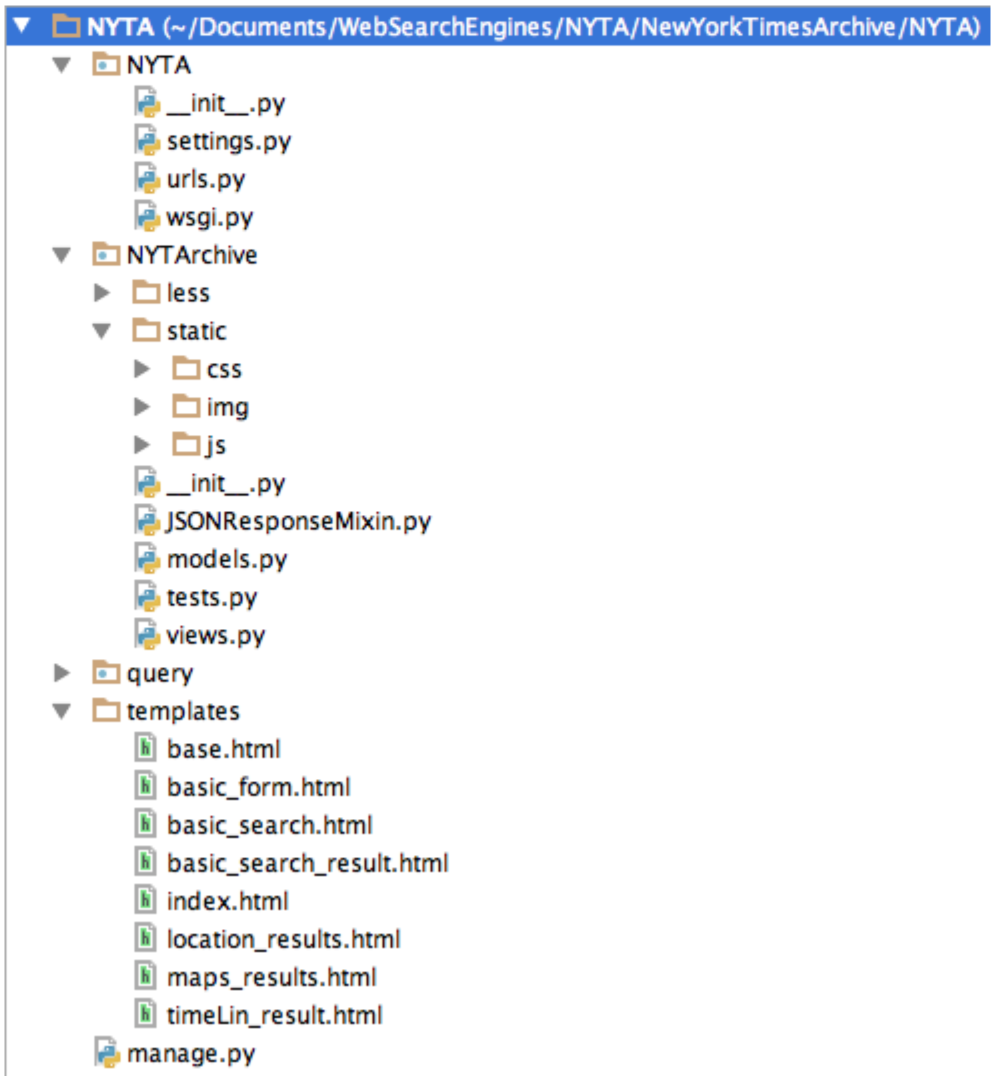
Figure 6

5. How to Implement a Search Engine Website

- **Python and Django**

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Django makes it easier to build better Web apps more quickly and with less code.

The following screenshot is a summary of our Django project for this system:



```
NYTA (~/Documents/WebSearchEngines/NYTA/NewYorkTimesArchive/NYTA)
    NYTA
        __init__.py
        settings.py
        urls.py
        wsgi.py
    NYTArchive
        less
        static
            css
            img
            js
        __init__.py
        JSONResponseMixin.py
        models.py
        tests.py
        views.py
    query
    templates
        base.html
        basic_form.html
        basic_search.html
        basic_search_result.html
        index.html
        location_results.html
        maps_results.html
        timeLin_result.html
    manage.py
```

- **Twitter Bootstrap and LESS**

For building Front-End User Interfaces, we use Twitter Bootstrap, a sleek, intuitive, and powerful front-end framework for faster and easier web development. Bootstrap utilizes LESS CSS, a dynamic stylesheet language. LESS extends CSS with dynamic behavior such as variables, mixins, operations and functions.

- **JavaScript, jQuery, HTML5, and CSS3**

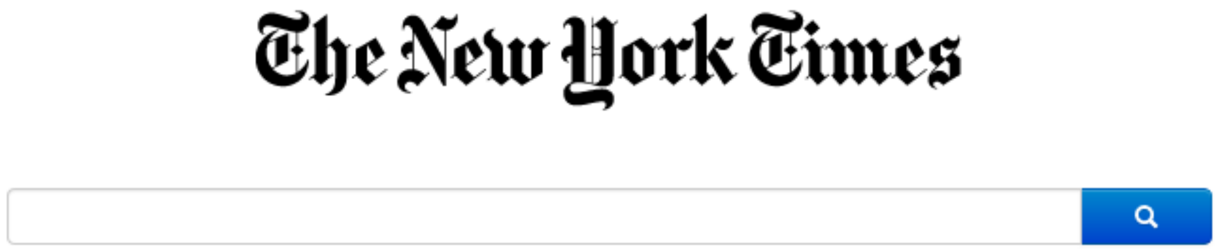
JavaScript, jQuery, HTML5 & CSS3 take our design to another level.

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

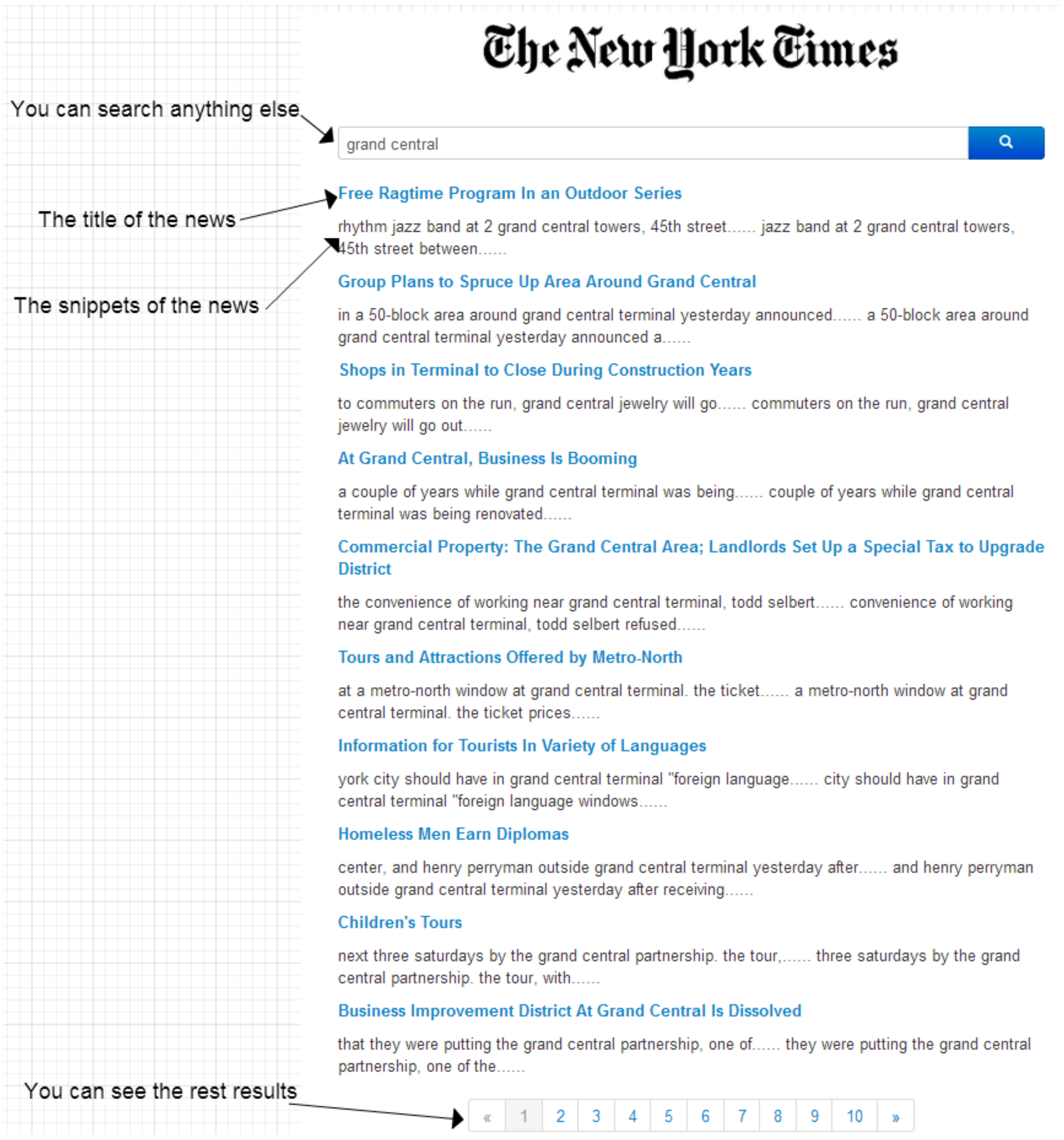
6. Search Newspapers

This is the basic function of this project, it will search the data set by keyword and return the result of the search.

Firstly, it will provide a very simple html for search as following:



The users can type any words they want to find out the news they need. And the result will be:



grand central

Free Ragtime Program In an Outdoor Series

rhythm jazz band at 2 grand central towers, 45th street...... jazz band at 2 grand central towers, 45th street between......

Group Plans to Spruce Up Area Around Grand Central

in a 50-block area around grand central terminal yesterday announced...... a 50-block area around grand central terminal yesterday announced a......

Shops in Terminal to Close During Construction Years

to commuters on the run, grand central jewelry will go...... commuters on the run, grand central jewelry will go out......

At Grand Central, Business Is Booming

a couple of years while grand central terminal was being...... couple of years while grand central terminal was being renovated......

Commercial Property: The Grand Central Area; Landlords Set Up a Special Tax to Upgrade District

the convenience of working near grand central terminal, todd selbert...... convenience of working near grand central terminal, todd selbert refused......

Tours and Attractions Offered by Metro-North

at a metro-north window at grand central terminal. the ticket...... a metro-north window at grand central terminal. the ticket prices......

Information for Tourists In Variety of Languages

york city should have in grand central terminal "foreign language...... city should have in grand central terminal "foreign language windows......

Homeless Men Earn Diplomas

center, and henry perryman outside grand central terminal yesterday after...... and henry perryman outside grand central terminal yesterday after receiving......

Children's Tours

next three saturdays by the grand central partnership. the tour,...... three saturdays by the grand central partnership. the tour, with......

Business Improvement District At Grand Central Is Dissolved

that they were putting the grand central partnership, one of...... they were putting the grand central partnership, one of the......

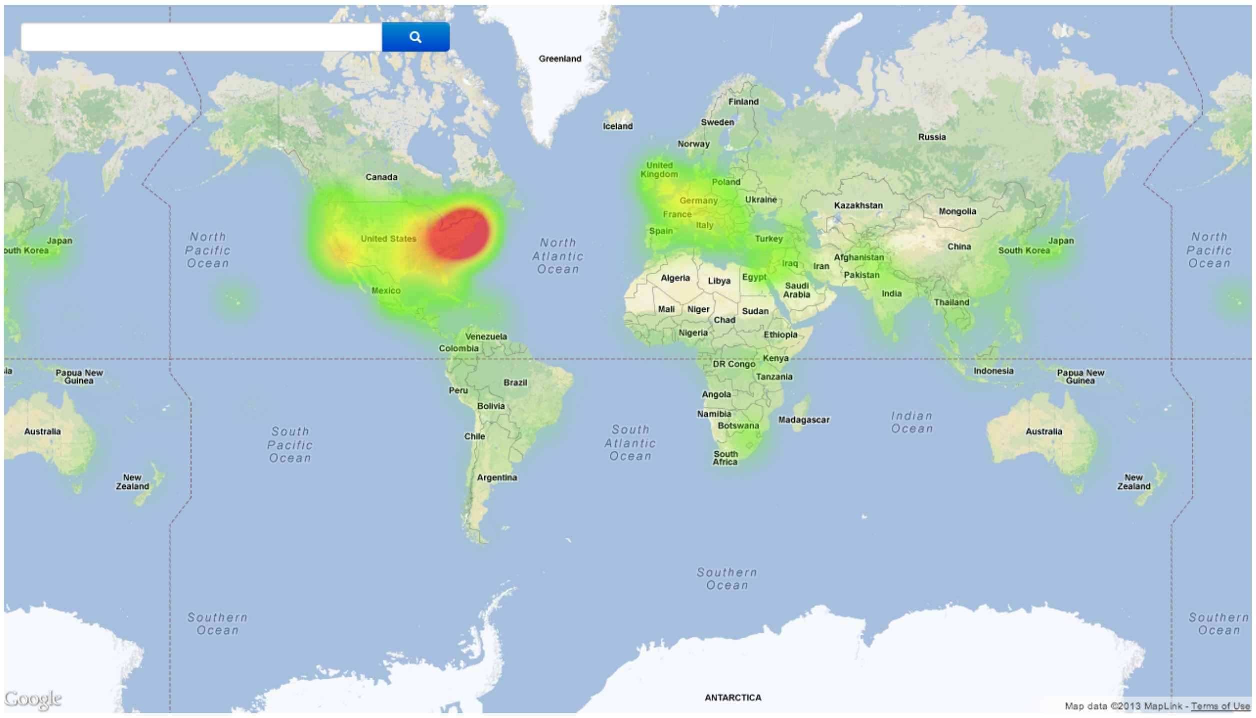
7. Browse Newspapers Using Google Maps

We extract geographic information from the articles, and then build this system that allows people to browse news using Google Maps.

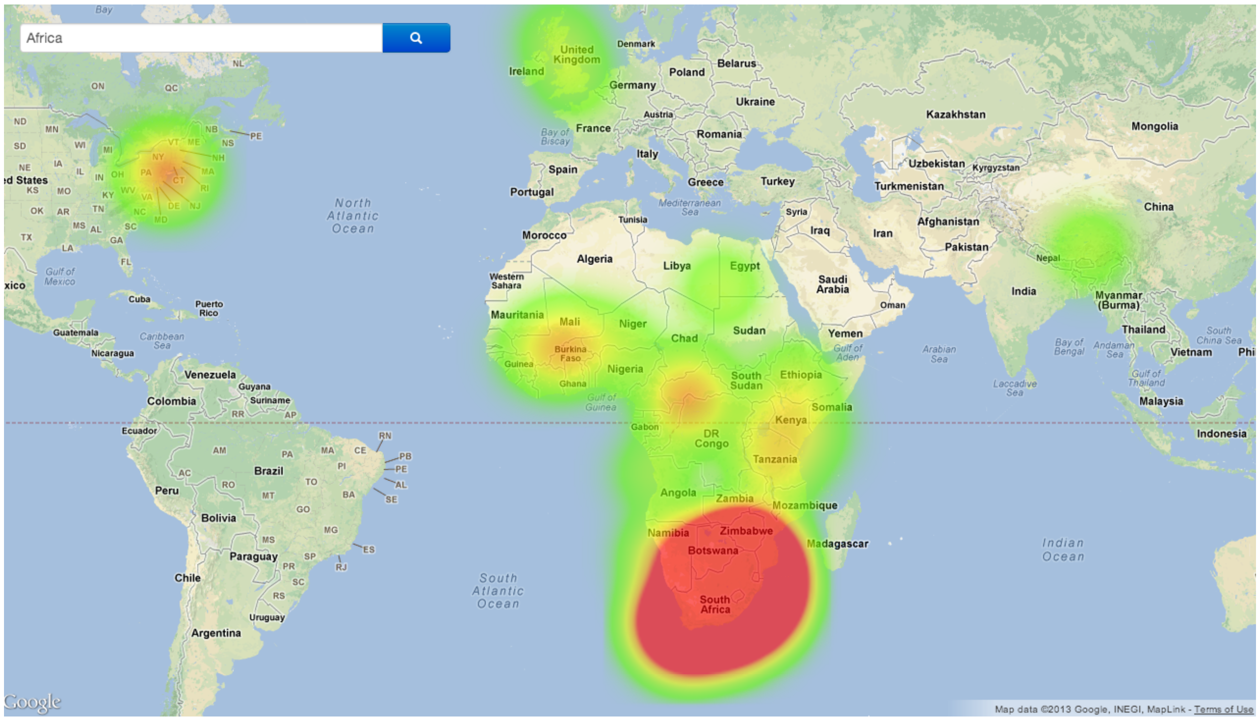
- **HeatMap (Statistical Analysis)**

The Heatmap Layer provides client side rendering of heatmaps. A heatmap is a visualization used to depict the intensity of data at geographical points. When the Heatmap Layer is enabled, a colored overlay will appear on top of the map. By default, areas of higher intensity will be colored red, and areas of lower intensity will appear green.

The following screenshot is the landing page for browsing newspapers using Google Maps, which shows the HeatMap of the intensity of locations mentioned in all the articles. As the HeatMap shows, locations in the east coast of the United States are mentioned the most in the New York Times newspaper articles between 1987 and 2007.

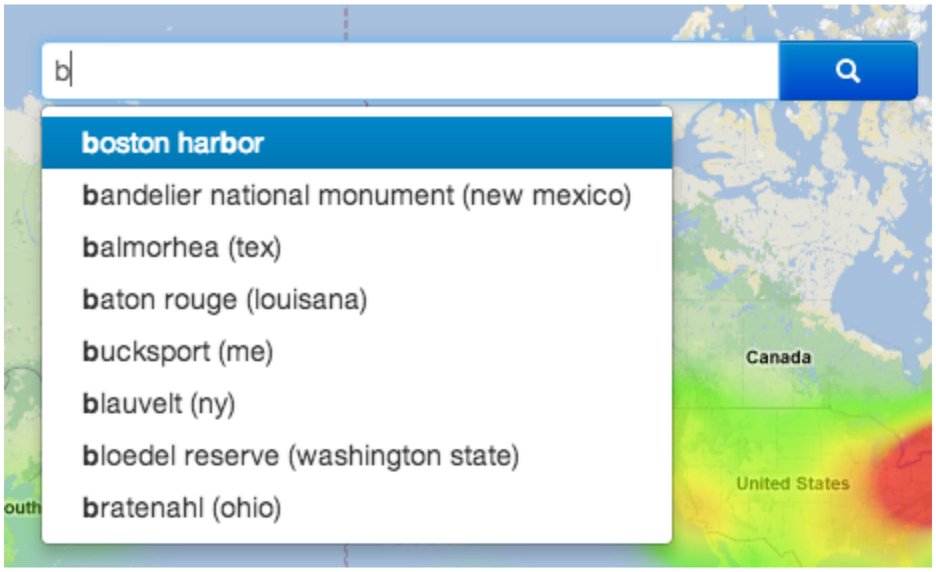


The HeatMap of searching “Africa”:



- **Typeahead**

A typeahead is a dropdown menu that appears when you're searching for something. It guesses what you're searching for so you can find it faster. If you see what you're looking for in the typeahead, click on it to save time. Typeahead is a fast and battle-tested functionality for auto completion.



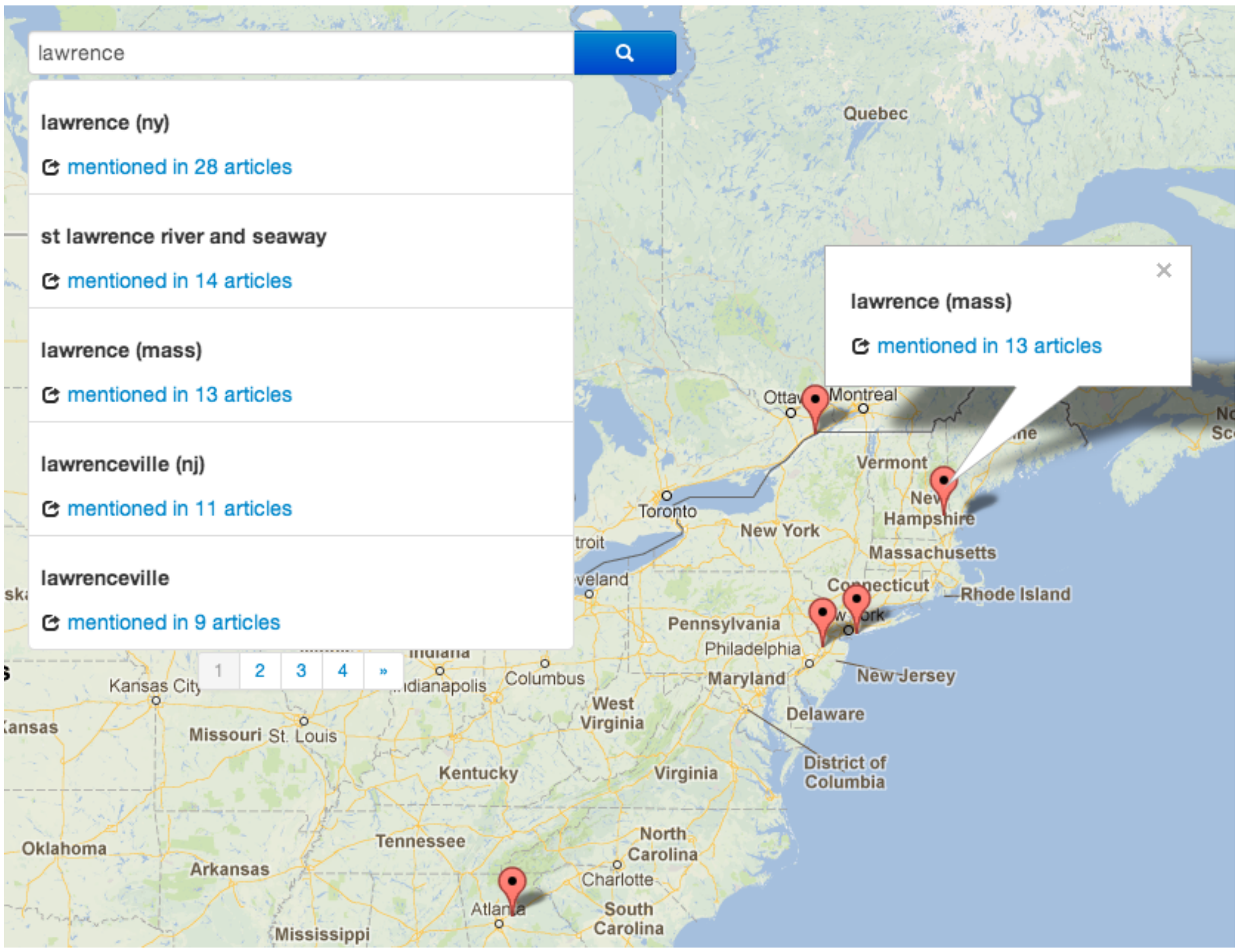
- **Dropdowns, Pagination, Markers, and InfoWindows**

You can search a location, and we'll use a dropdown list to show the search results. The dropdown list will be displayed with a sliding down motion. The result locations are shown in the descending order of the number of articles which mentioned the location.

If there're more than 5 locations in the search result, we'll use pagination to show the result list. Pagination is great for apps and search results. The large block is hard to miss, easily scalable, and provides large click areas.

Also, there will be markers on Google Maps to identify the exact location (latitude and longitude) of the search results. Markers are designed to be interactive. By default, they receive 'click' events, for example, and are used within event listeners to bring up InfoWindows. In the meantime, the marker will be animated ("bounce" in place) so that it exhibits dynamic movement in this circumstance.

InfoWindow displays content in a floating window above the map. The info window looks a little like a comic-book word balloon; it has a content area and a tapered stem, where the tip of the stem is at a specified location on the map. You can see the info window in action by clicking markers on Google Maps. InfoWindows will be attached to Marker objects (in which case its position is based on the marker's location).



lawrence

lawrence (ny)
mentioned in 28 articles

st lawrence river and seaway
mentioned in 14 articles

lawrence (mass)
mentioned in 13 articles

lawrenceville (nj)
mentioned in 11 articles

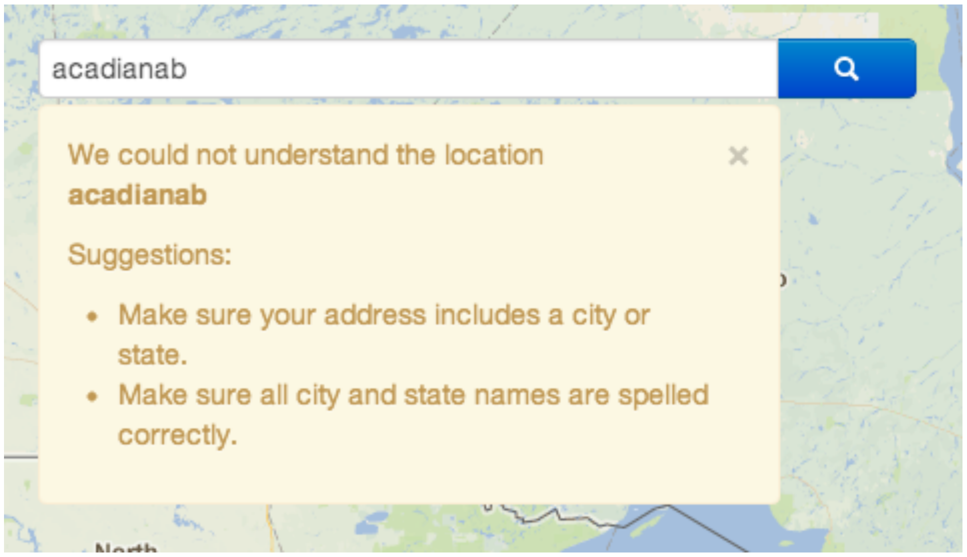
lawrenceville
mentioned in 9 articles

lawrence (mass)
mentioned in 13 articles

We enable a hover state on rows within the dropdown list and automatically change the center of the map to the latitude and longitude of the specific location (the location of the corresponding marker on Google Maps). The transition will be smoothly animated and the marker will be animated ("bounce" in place) so that you can easily notice dynamic movement and identify the location on Google Maps.

- ## Alerts

Styles for warning, and error messages.


acadianab

We could not understand the location **acadianab**

Suggestions:

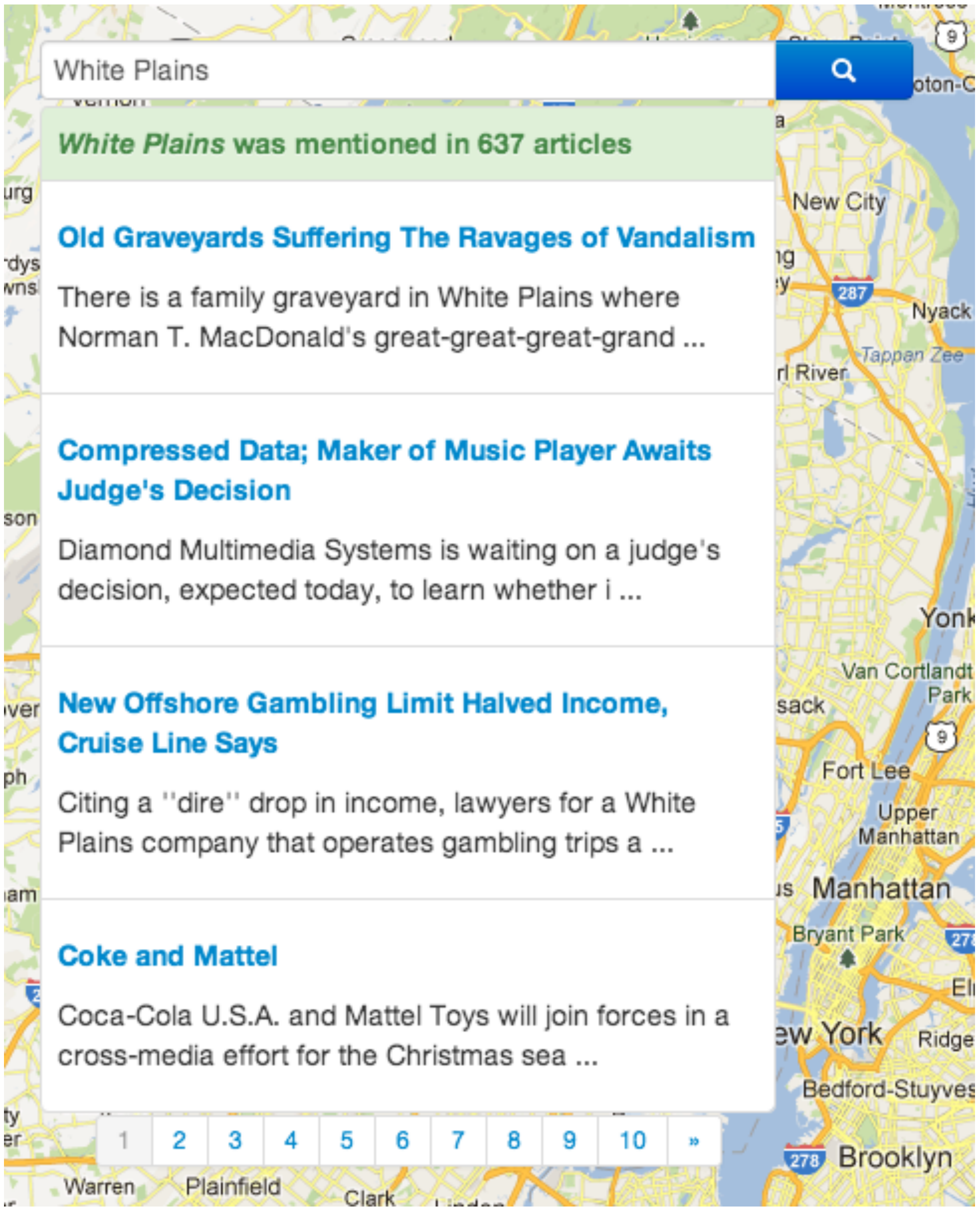
- Make sure your address includes a city or state.
- Make sure all city and state names are spelled correctly.

- **Browse Newspaper Articles**

You can browse newspaper articles by clicking the link ("mentioned in N articles") either in the dropdown list or in the InfoWindow. The original dropdown list of locations will be hidden and a new dropdown list of articles will be slided down, in which we'll list the articles that mentioned this specific location. If there're more than 4 articles in the search results, we'll use pagination to show the result list.

In the head of the dropdown list of articles, we'll show you the summary information: "*LOCATION* was mentioned in N articles" (in the styles for information messages), to give you a sense of the total amount of articles in the dropdown list, while you browse articles using pagination.

In each row of article dropdown list, we'll show you the headline of the article and two lines of lead paragraph snippets. After you click the headline link, a new window will be opened showing the online edition of that article on nytimes.com for you.


White Plains

White Plains was mentioned in 637 articles

Old Graveyards Suffering The Ravages of Vandalism

There is a family graveyard in White Plains where Norman T. MacDonald's great-great-great-grand ...

Compressed Data; Maker of Music Player Awaits Judge's Decision

Diamond Multimedia Systems is waiting on a judge's decision, expected today, to learn whether i ...

New Offshore Gambling Limit Halved Income, Cruise Line Says

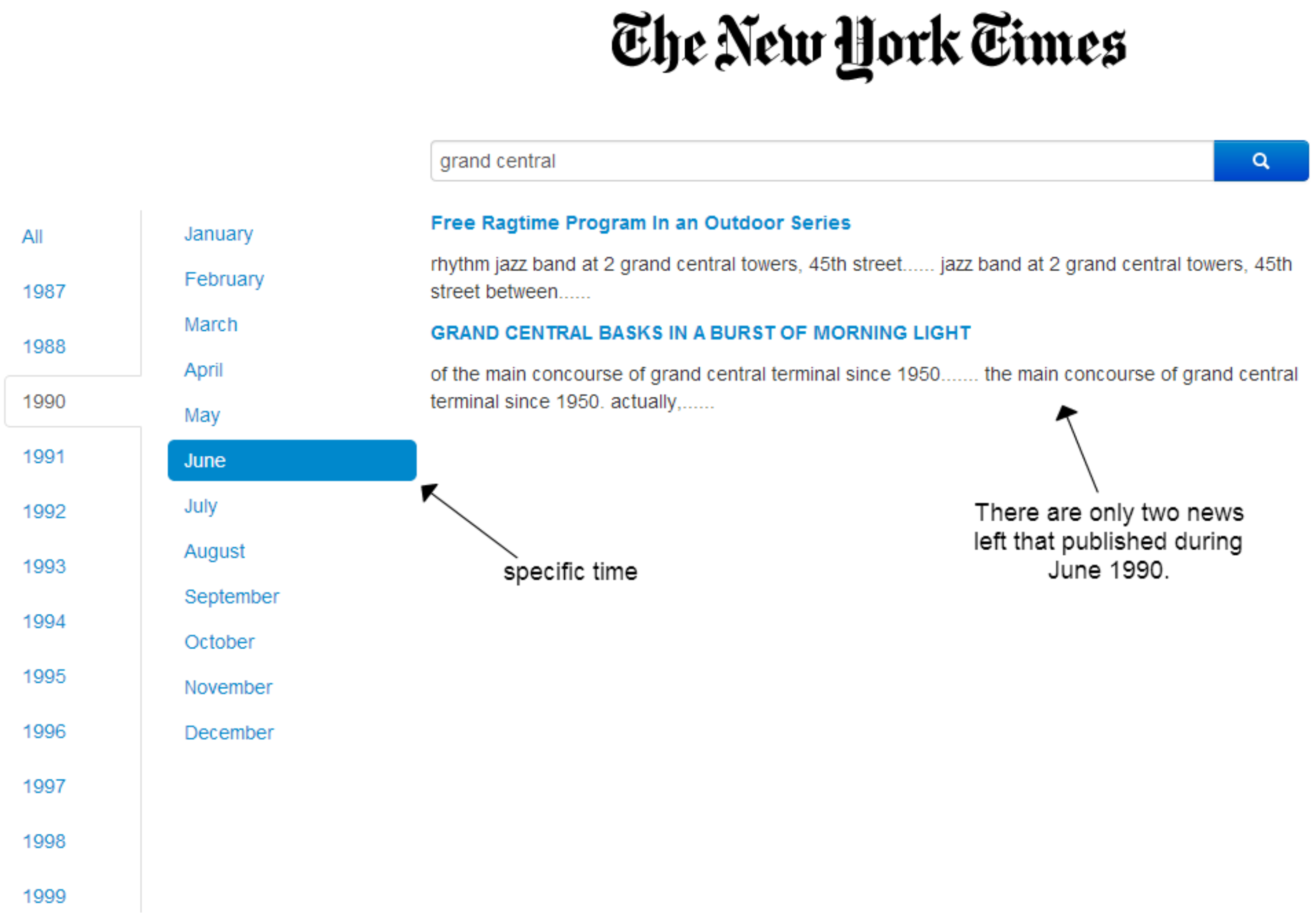
Citing a ''dire'' drop in income, lawyers for a White Plains company that operates gambling trips a ...

Coke and Mattel

Coca-Cola U.S.A. and Mattel Toys will join forces in a cross-media effort for the Christmas sea ...

8. Browse Newspapers Using Timeline

This is an interesting feature in our project, in the part, the users can filter the search result by telling the website these news should be published during a specific time.



Free Ragtime Program In an Outdoor Series

rhythm jazz band at 2 grand central towers, 45th street...... jazz band at 2 grand central towers, 45th street between......

GRAND CENTRAL BASKS IN A BURST OF MORNING LIGHT

of the main concourse of grand central terminal since 1950....... the main concourse of grand central terminal since 1950. actually,......

The timeline function is implemented by Ajax, once you choose the year and month, the result will be filtered. For example, in the section 6, we get the results of all news about “grand central”. If we choose “1990” and “June”, the results will be filtered and the left are all published in June 1990.

9. Search In Action: Experiment

How long does it take to search on provided data set?

The time cost really depends on the query. For some special query, like very common words “for, the”, it would be as slow as several seconds (3s at most for the largest cases “for” and “the”), although we have already cache them in memory (cache decoded data for these words). For regular cases like “Beijing, New York City, Poly”, the speed would be as fast as micro seconds level. For example, the “iloveyou” query uses 0.003 s.

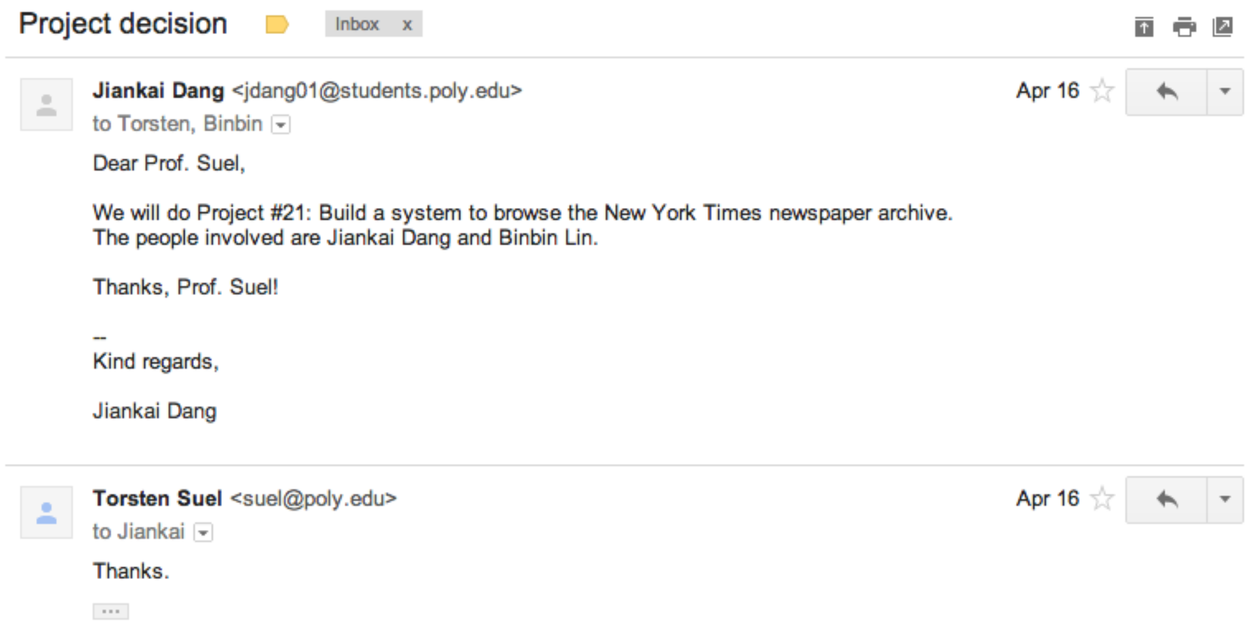
References

[1] http://en.wikipedia.org/wiki/interted_index
[2] http://nlp.stanford.edu/IR-book/html/htmledition/variable-byte-codes-1.html
[3] http://en.wikipedia.org/wiki/Okapi_BM25
[4] http://www.datatang.com/data/10131
[5] http://nltk.org/install.html

Appendix

Email:
jdang01@students.poly.edu
bl1402@students.poly.edu

Screenshot of the email we sent you (this makes it even easier for you to check things :)



Project decision Inbox x

Jiankai Dang <jdang01@students.poly.edu> Apr 16
to Torsten, Binbin

Dear Prof. Suel,

We will do Project #21: Build a system to browse the New York Times newspaper archive.
The people involved are Jiankai Dang and Binbin Lin.

Thanks, Prof. Suel!

--
Kind regards,

Jiankai Dang

Torsten Suel <suel@poly.edu> Apr 16
to Jiankai

Thanks.