



# NewsRoom

[Related news articles and social media posts for Wikipedia]  
Project Report

## [Team Members]

- |                            |             |
|----------------------------|-------------|
| 1. Ananda Kanagaraj Sankar | - #50133315 |
| 2. Karthik Janarthanan     | - #50133543 |
| 3. Saravanan Adaikkalavan  | - #50134295 |
| 4. Naveen Kumar Ramamurthy | - #50134555 |

## Table of Contents

Introduction .....	3
System Overview .....	3
Overall System .....	3
Wikipedia Indexing.....	4
News/Tweets Fetch and Indexing Module .....	4
Query Processing Module .....	4
Features .....	5
Role of Solr .....	6
Solr Schema.....	10
Field Types .....	11
Third Party Libraries .....	12
Solr Stats .....	13
Index.....	13
Cache.....	13
Team Contribution .....	14
Ananda Kanagaraj Sankar .....	14
Karthik Janarthanan .....	14
Saravanan Adaikkalavan .....	14
Naveen Kumar Ramamurthy.....	14
Future Work.....	14
Screenshots.....	14

## Introduction

Wikipedia is now the Web's third-most-popular news and information source, beating the sites of CNN and Yahoo News, according to Nielsen NetRatings. Many of the digital organizations in the digital native news world however now emphasize the importance of social media in storytelling and engaging their audiences. The aim of the NewsRoom therefore is to build a two-tiered search system – one that serves Wikipedia articles and the other that provides related news stories. The idea is to be able to serve **contextually related** news stories and the relevant posts on the social media platforms for the given Wikipedia article(s). The NewsRoom therefore has been built to serve the users that seek informational need such as news with the following results –

- Wikipedia articles,
- the contextually related different news stories from The New York Times and
- a glimpse of a set of associated tweets for the Wikipedia article(s)

in one place. In real-time, the concept of aggregation designed as part of the NewsRoom helps to integrate the information from various different sources onto a single platform and thus reduces the time and effort required to look for a piece of information on many websites, creating a unique information space to the user. This document provides a high level description of all the functions, specifications and the implementation details of the NewsRoom search system.

## System Overview

This section describes major components of the system, their interconnections and external interfaces.

### Overall System

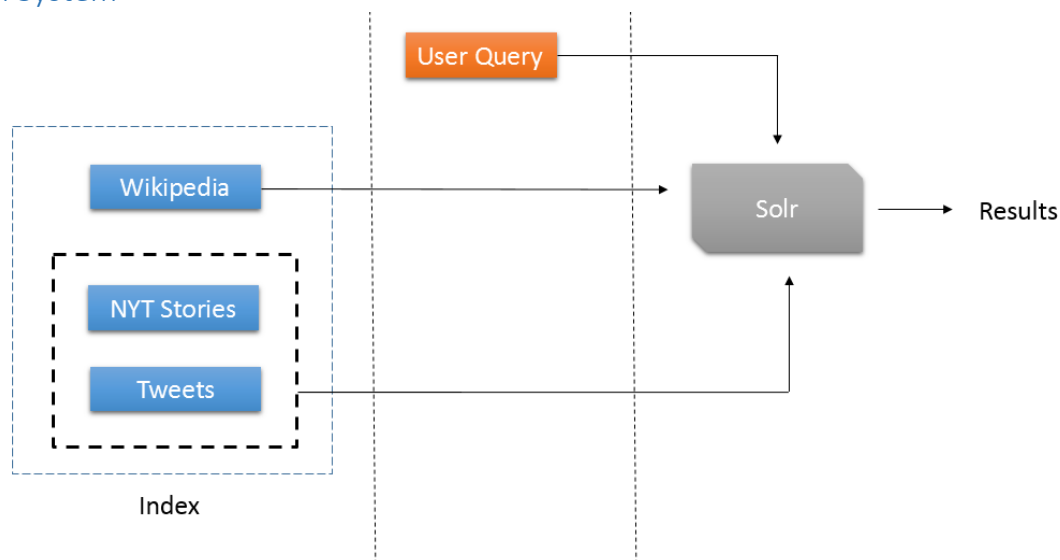


Figure 1 Overall System

## Wikipedia Indexing

The latest dataset of over 200 thousand Wikipedia articles is obtained from the Wikimedia repository - <http://dumps.wikimedia.org/enwiki/latest/>. The corpus of XML documents are then parsed, processed and indexed using the Solr system.

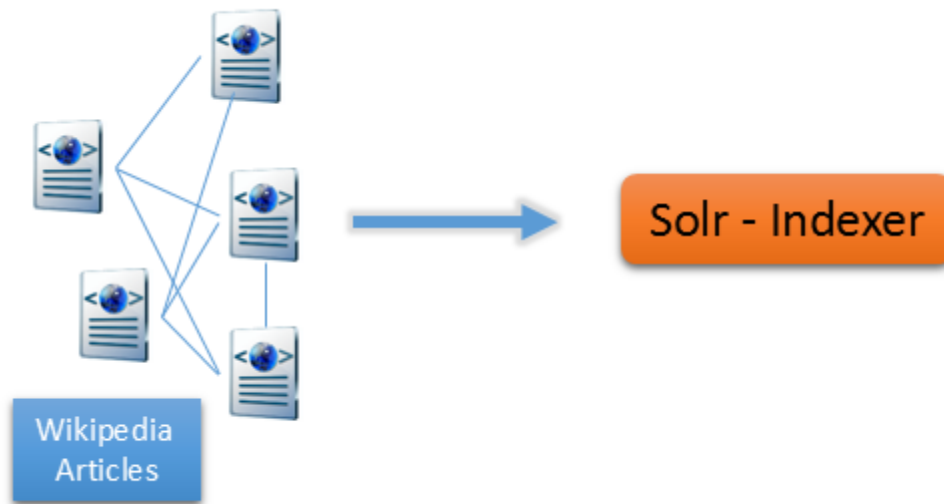


Figure 2 Wikipedia Module

## News/Tweets Fetch and Indexing Module

The New York Times 2007 News dataset that was provided is parsed in a similar fashion as the Wikipedia documents and indexed using the SOLR system in the above described way. The Twitter dataset that was created by Twitter Sentiment Classification using Distant Supervision at the Stanford University is used to index the user tweets.

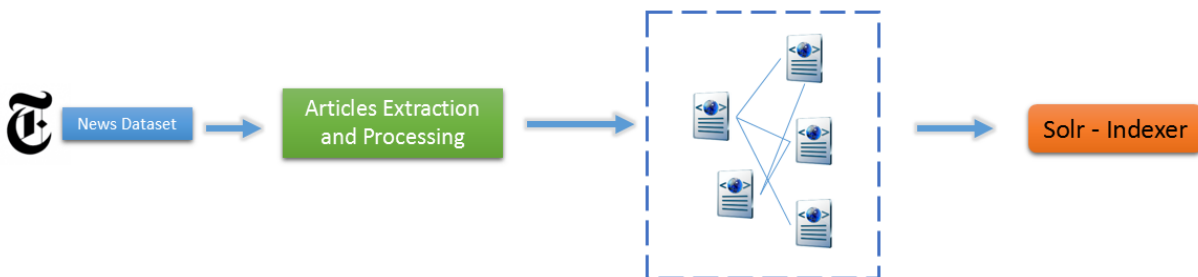


Figure 3 News/Tweets Module

## Query Processing Module

The given user query is processed and passed to the Solr system to retrieve relevant Wikipedia articles. The retrieved documents are logically ranked and displayed in the UI. After the user selects the desired document from the results, the selected result is passed to the results processor that sends a request to the Solr system to retrieve the related tweets and the news articles. The processed results are then displayed in the UI.

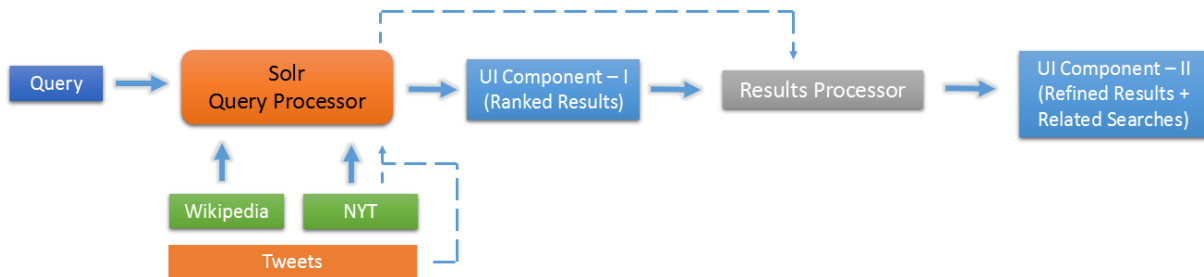


Figure 4 Query Processing Module

## Features

- Newsroom serves the Wikipedia articles, the contextually related news stories from The New York Times and the related tweets, all in one place.
- **Dynamic Indexing On the Go**  
Newsroom does not intend to disappoint the users when the existing Index does not contain the documents that the users are looking for. The concept of 'Near' real-time indexing is incorporated in such a way that the system queries the original source of Wikipedia articles and indexes them so that they are readily available for search after being indexed.
- **Spell-Checker**  
To ease the users' efforts to know the correct spelling of the terms being queried, the Newsroom's "*Did you mean?*" feature provides the alternate correct spelling suggestions considering the phonetics of the terms. The users shall be able to provide the correct intended search query by merely choosing the best option from the listed spelling suggestions in case of a faulty query input. An interactive spelling correction facility that presents possible appropriate corrections to their erroneous queries would improve the system in terms of precision, recall and user-effort.
- **Auto-Complete**  
The system provides search predictions that might sound similar to the search terms dynamically in real-time as the user types in the search box.
- **Hot Trends and Content Tags**  
The user shall get to see the trending categories across the medium and a list of important keywords extracted out of the content that's currently viewed. The user can also continue to search based on these keywords by just clicking on them.
- **Related Wikipedia Articles**  
In addition to the related tweets and the news stories, the users are provided with the links to the related Wikipedia articles as well.

- **Relevancy Percentage**

The users shall know the measure of how relevant the document, that is currently being viewed, to the search query in terms of a percentage on the UI.

## Role of Solr

1. **Create Index**

The Wikipedia dump containing articles is parsed using PHP and are added to index through the Solarium, the PHP Solr client. In addition, the articles retrieved dynamically during the user query are also parsed and added to the index through Solarium.

2. **Fetch Relevant Wikipedia Articles**

The Lucene's standard scoring mechanism of using a combination of the Vector Space Model (VSM) of Information Retrieval and the Boolean model is used to determine how relevant a Wikipedia document is to a User's query.

3. **Fetch Related News Stories and Tweets**

The related News Stories and the related Tweets sections which show content based on the Wikipedia article that is currently chosen are built using the *MoreLikeThis (MLT)* class. The majority of the work is done by analyzing the content of the Wikipedia article (source document) relative to the aggregate of all the news stories content that exists in the index. The MLT request handler is configured to match on the title and the content information and calculate the 'relatedness' based on the term-frequency and the document-frequency thresholds as shown below. The score implicitly computed by the MLT is then considered to perform term-boosting. The importance provided is high for the match with the field 'title' rather than the field 'newsContent' as it is assumed that the gist of the news story will logically be covered as part of the title by itself.

```
<requestHandler name="/mlt" class="solr.MoreLikeThisHandler">
  <lst name="defaults">
    <str name="mlt.fl">title,newsContent</str>
    <str name="mlt.mintf">1</str>
    <str name="mlt.mindf">2</str>
    <str name="mlt.boost">true</str>
    <str name="mlt.qf">title^4 newsContent^2</str>
    <str name=" "></str>
  </lst>
</requestHandler>
```

Figure 5 MoreLikeThis Schema Definition

As explained above, the 'relatedness' for the tweets is similarly computed based on the content of the tweets in the Index.

#### 4. NearRealTime Indexing

The NearRealTime Indexing feature of Solr is employed when the system returned zero results for a query. When no Wikipedia articles are obtained from the existing Index for a user query, the related articles are fetched from the actual Wikipedia source by means of re-creating a query instance with the help of the [Wikimedia API](#). The identified articles are then added to the existing index in run-time so that they are available for search immediately after being indexed. These results are eventually displayed to the end-user.

#### 5. Highlighter

The Lucene's Standard Highlighter is used to highlight the search keywords in the search snippets. The highlighting component is obtained as configured in Lucene and the settings are applied as shown below for an instance.

```
<!-- Highlighting defaults -->
<str name="hl">on</str>
<str name="hl.fl">content title</str>
<str name="hl.preserveMulti">true</str>
<str name="hl.encoder">html</str>
<str name="hl.simple.pre">&lt;b></str>
<str name="hl.simple.post">&lt;/b></str>
<str name="f.title.hl.fragsize">100</str>
<str name="f.title.hl.alternateField">title</str>
<str name="f.content.hl.snippets">3</str>
<str name="f.content.hl.fragsize">300</str>
<str name="f.content.hl.alternateField">content</str>
<str name="f.content.hl.maxAlternateFieldLength">750</str>
```

Figure 6 Highlighter definition in query schema

```

<searchComponent class="solr.HighlightComponent" name="highlight">
  <highlighting>
    <!-- Configure the standard fragmenter -->
    <!-- This could most likely be commented out in the "default" case -->
    <fragmenter name="gap" default="true" class="solr.highlight.GapFragmenter">
      <lst name="defaults">
        <int name="hl.fragsize">100</int>
      </lst>
    </fragmenter>

    <!-- Configure the standard formatter -->
    <formatter name="html" default="true" class="solr.highlight.HtmlFormatter">
      <lst name="defaults">
        <str name="hl.simple.pre"><![CDATA[<em>]]></str>
        <str name="hl.simple.post"><![CDATA[</em>]]></str>
      </lst>
    </formatter>

    <!-- Configure the standard encoder -->
    <encoder name="html" class="solr.highlight.HtmlEncoder" />

    <!-- Configure the standard fragListBuilder -->
    <fragListBuilder name="simple" class="solr.highlight.SimpleFragListBuilder"/>

    <boundaryScanner name="default" default="true" class="solr.highlight.SimpleBoundaryScanner">
      <lst name="defaults">
        <str name="hl.bs.maxScan">10</str>
        <str name="hl.bs.chars">.,!? \t;@#10;13;</str>
      </lst>
    </boundaryScanner>
  </highlighting>
</searchComponent>

```

Figure 7 Highlighter SearchComponent

## 6. SpellChecker

When the user has misspelled a search phrase, “Did you mean” feature of the NewsRoom will list the most likely alternative for the misspelled phrase. For this feature SpellCheckerComponent of Solr is implemented as a search component with the default Lucene SpellChecker class. As the DirectSolrSpellChecker has been used, the main Solr index will be used for spellcheck. The suggestions for spellcheck are sorted based on the frequency of the word in the index. The field ‘textspell’, which is the copyfield of field ‘content’, in the schema is used for the purpose of spellcheck indexing.

The configuration for the spellchecker in solrconfig.xml is shown below.



```

<lst name="spellchecker">
  <str name="name">default</str>
  <str name="field">textSpell</str>
  <str name="comparatorClass">freq</str>
  <str name="accuracy">0.7</str>
  <int name="maxEdits">2</int>
  <int name="minPrefix">1</int>
  <int name="maxInspections">5</int>
  <int name="minQueryLength">4</int>
  <float name="maxQueryFrequency">0.01</float>
</lst>

```

Figure 8 SpellChecker definition in query schema

```

<searchComponent name="spellcheck" class="solr.SpellCheckComponent">

  <str name="queryAnalyzerFieldType">textSpell</str>

  <!-- a spellchecker built from a field of the main index -->
  <lst name="spellchecker">
    <str name="name">default</str>
    <str name="field">textSpell</str>
    <str name="comparatorClass">freq</str>
    <str name="accuracy">0.7</str>
    <int name="maxEdits">2</int>
    <int name="minPrefix">1</int>
    <int name="maxInspections">5</int>
    <int name="minQueryLength">4</int><float name="maxQueryFrequency">0.01</float>
  </lst>

</searchComponent>

```

Figure 9 SpellChecker SearchComponent

## 7. Auto-Complete

As the user types in the NewsRoom search box, the user can find information quickly by seeing search predictions that might be similar to the search terms that are being typed. When the user clicks a search prediction, he/she do a search using that prediction as the search term. The search queries that are seen as part of Autocomplete are a reflection of the title and the content of the documents indexed. The SuggestComponent of Solr is used to provide users with the automatic suggestions for the query terms. This is implemented by adding a search component in solrconfig.xml as shown below to extend spell checker.

```

<searchComponent name="suggest" class="solr.SuggestComponent">
  <lst name="suggester">
    <str name="name">mySuggester</str>
    <str name="lookupImpl">FuzzyLookupFactory</str>
    <str name="dictionaryImpl">DocumentDictionaryFactory</str>
    <str name="field">title</str>
    <str name="weightField">authorid</str>
    <str name="suggestAnalyzerFieldType">string</str>
  </lst>
</searchComponent>

```

Figure 10 Autocomplete Schema Definition

## 8. Tags and Keywords

The [MoreLikeThis](#) handler provided by Lucene is used to fetch interesting terms from the Wikipedia article to display the most important set of keywords to the user as shown below.

```

$wikiresultset = $this->client->select($query);
$tags=$wikiresultset->getInterestingTerms();

```

Figure 11 Snippet to display the most interesting keywords

## Solr Schema

The raw documents that are obtained in the Wikipedia and The New York Times datasets need to be processed and parsed. The system however requires a set of essential fields off the documents to process and Index them. The following are the fields that are outlined in the Solr Schema.

### Wikipedia

Field Name	Functionality
<b>revision</b>	Identity semantic of the document.
<b>content</b>	Text of the Wikipedia article.

### The New York Times

Field Name	Functionality
<b>person</b>	People associated with the news story.
<b>category</b>	Category that the article falls under, eg - Law Enforcement & Security.
<b>link</b>	Webpage link of the article on The New York Times
<b>source</b>	Source of the news article.
<b>newsContent</b>	Text of the news article.

## Twitter

Field Name	Functionality
tweetContent	Text of the tweet.

## Shared Fields

Field Name	Functionality
id	Identity semantic of the document.
author	Writer of the Wikipedia article/News Story/Tweet.
authored	Identity semantic of the author.
timestamp	Date of publication of the article/Date of the Tweet posted.
title	Title of the article, news story.

**Unique Key** – The id of the document is set to be unique key.

## Field Types

The significant fields such as the title and the content of the Wikipedia article, NYT news article and the tweet instance are configured to be of type `text_general`. The analyzer chains are configured to examine the content of the fields set as `text_general` and generate the token stream at the time of building the Index and querying. The analyzer is constructed by chaining a tokenizer and a set of token filters as listed below

## At Indexing Time

```
<analyzer type="index">
  <tokenizer class="solr.ClassicTokenizerFactory"/>
  <filter class="solr.WordDelimiterFilterFactory"
    generateWordParts="0"
    generateNumberParts="0"
    catenateWords="0"
    catenateNumbers="0"
    catenateAll="0"
    preserveOriginal="1"
  />
  <filter class="solr.LowerCaseFilterFactory"/>
  <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
  <filter class="solr.PorterStemFilterFactory"/>
  <filter class="solr.TrimFilterFactory"/>
  <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
</analyzer>
```

Figure 12 Analyzers and Filters in Schema for Indexing

As shown above, the [ClassicTokenizer](#) is used to split the text\_general field into tokens, treating whitespace and punctuation as delimiters.

### Chain of Filters

- WordDelimiterFilter – The tokens are split into sub-tokens with the exceptions like - the original token to be indexed is preserved without modifications (in addition to the tokens generated due to other options); the parts of the word shall not be tokenized when camelcased; the numerals shall not be split into sub-tokens when they're of the form 456-67.
- LowerCaseFilter – To lowercase the letters in each token.
- StopFilter – To discard common words. A customized stop word list "stopwords.txt" has been specified with the "words" attribute in the schema. The "ignoreCase" attribute is used to ignore the case of tokens when comparing to the stopword list.
- PorterStemFilter – This filter implements Porter's stemmer strategies of reducing the word such as 'walks', 'walking' to its elemental root like 'walk'.
- TrimFilter – To trim the whitespace at either end of the token.
- RemoveDuplicatesTokenFilter – To filter out any tokens in the tokenstream with the same text. This filter is chained after stemming because stemming synonyms with similar roots causes duplicates in the tokenstream.

### At Querying Time

```
<analyzer type="query">
  <tokenizer class="solr.StandardTokenizerFactory"/>
  <filter class="solr.LowerCaseFilterFactory"/>
  <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />
  <filter class="solr.PorterStemFilterFactory"/>
</analyzer>
```

Figure 13 Analyzers and Filters in Schema for query

[StandardTokenizer](#) is used to tokenize the field comprising of the LowerCaseFilter, StopFilter and PorterStemFilter as explained above.

### Third Party Libraries


The following is the list of third party libraries used in the NewsRoom system.

- *Solarium* library for PHP Solr client to access the solr system.
- *Jquery* and other javascript libraries for UI

## Solr Stats

### Index

Over 1 million articles are indexed totally in NewsRoom, with a total of 108507 Wikipedia articles, 39953 News articles and 1597381 Twitter posts. The size of the index comes around 1.78 Gigabytes. The stats of the index is shown below.



## Statistics

Last Modified: 5 days ago

Num Docs: 1745841

Max Doc: 1785796


Heap Memory 4674352


Usage:

Deleted Docs: 39955

Version: 1119

Segment Count: 25

Current: 



## Replication (Master)

	Version	Gen	Size
Master (Searching)	1417393509707	329	1.78 GB

Figure 14 Index Stats

### Cache

The following is the stats for the cache in solr system of NewsRoom.

documentCache		
class:	org.apache.solr.search.LRUCache	
version:	1.0	
description:	LRU Cache(maxSize=512, initialSize=512)	
src:	null	
stats:	lookups:	20
	hits:	8
	hitratio:	0.4
	inserts:	12
	evictions:	0
	size:	10
	warmupTime:	0
	cumulative_lookups:	20
	cumulative_hits:	8
	cumulative_hitratio:	0.4
	cumulative_inserts:	12
	cumulative_evictions:	0

Figure 15 Cache stats

## Team Contribution

### Ananda Kanagaraj Sankar

Developed Wikipedia dump parser, Solr schema for Wiki articles, Solr Highlighter, dynamic retrieval and real-time indexing of Wikipedia articles, backend using Solarium PHP Solr client, UI Components and Video documentation.

### Karthik Janarthanan

Developed dump parser for News articles, Solr schema for news articles, Solr MoreLikeThis for News articles, Solr SpellCheck and suggestions, Solr analyzers and filters schema for index and query and documentation.

### Saravanan Adaikkalavan

Developed dump parser for Twitter posts, Solr schema for Twitter posts, Solr MoreLikeThis for Twitter posts, Solr fields schema, UI components and documentation.

### Naveen Kumar Ramamurthy

Developed Solr schema for Twitter posts, Solr AutoComplete feature, Solr MoreLikeThis for Wiki articles, Hot trends and Top Keywords extraction using Solr and Video documentation

## Future Work

Future work and improvements for NewsRoom includes dynamic retrieval and indexing for news articles from various sources and tweets from Twitter. Due to restrictions and limitations on Twitter API, dynamic retrieval is not yet implemented. For importing data from dump into the index, currently, a parser is developed in PHP and Solarium client is used for indexing. To improve the performance of indexing, data import handler can be designed in such a way to parse and index the documents from large sized dump sources.

## Screenshots

Below are the different screens of the NewsRoom system.

- Search Results for the query “*Microsoft*”. Search results are displayed with the image present in the article enclosed within <img> tags.

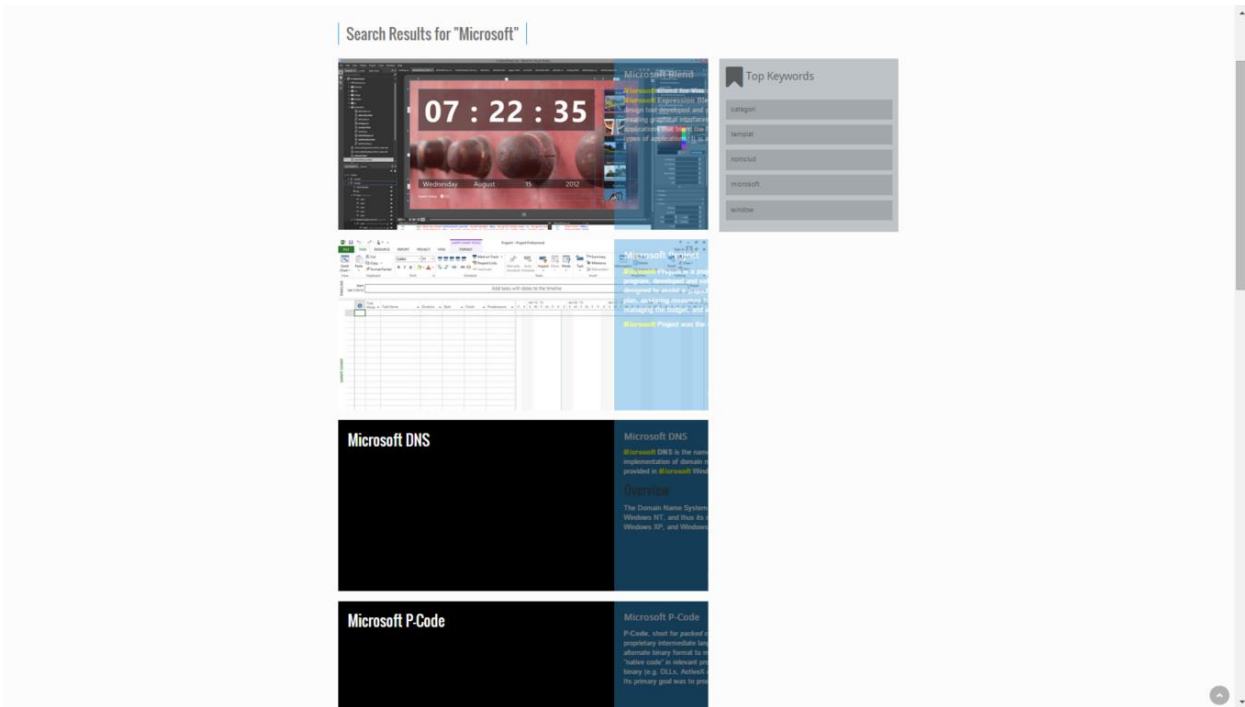


Figure 16 Search Results

- SpellCheck feature is used and the suggestions for the misspelled query “Rajinikan” is displayed.

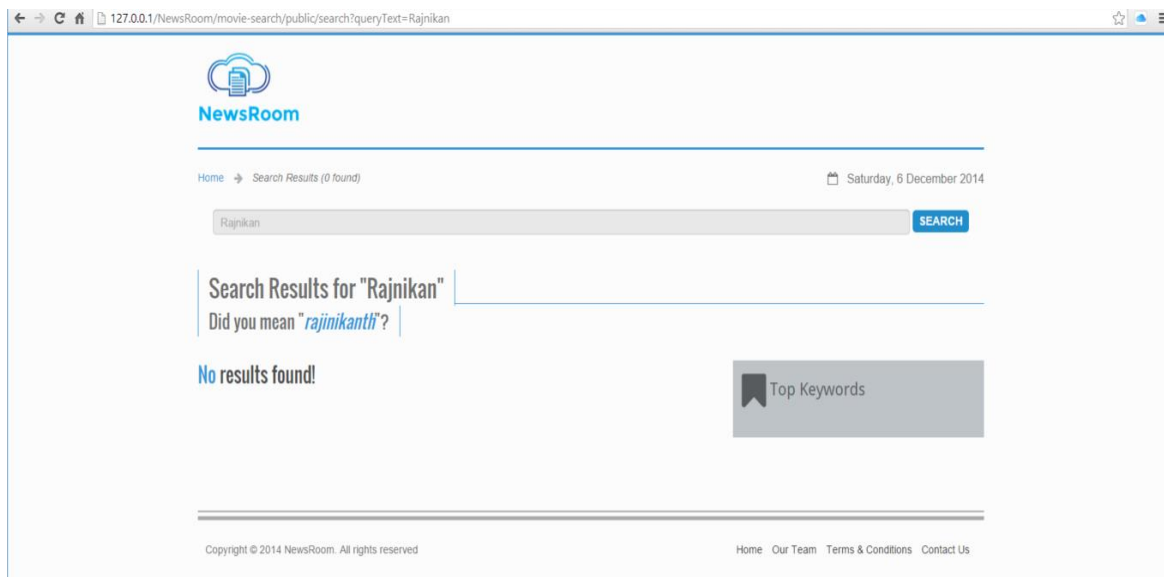


Figure 17 SpellCheck Auto-suggest feature

- AutoComplete suggestions start displaying when the user starts typing the query.

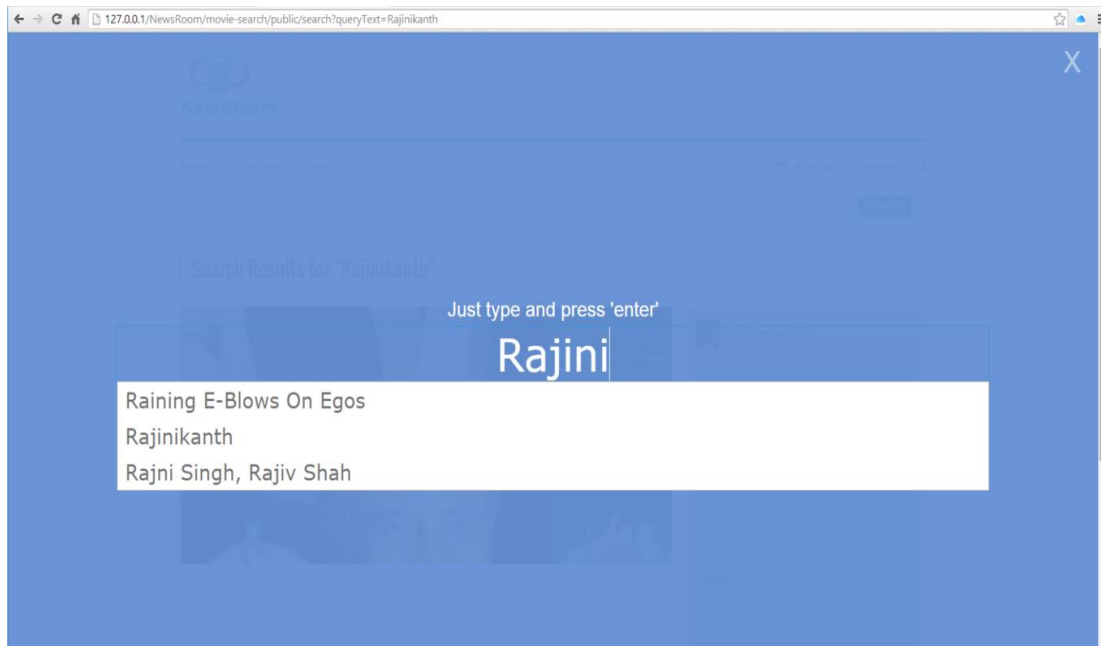


Figure 18 AutoComplete feature

- The article about “Sachin Tendulkar” is not present in the existing index. Hence, when such a query is obtained, the article is dynamically retrieved and indexed and is shown to the user. Also the related keywords are displayed on the right side of the screen. The query terms are highlighted in the snippet shown in the results.

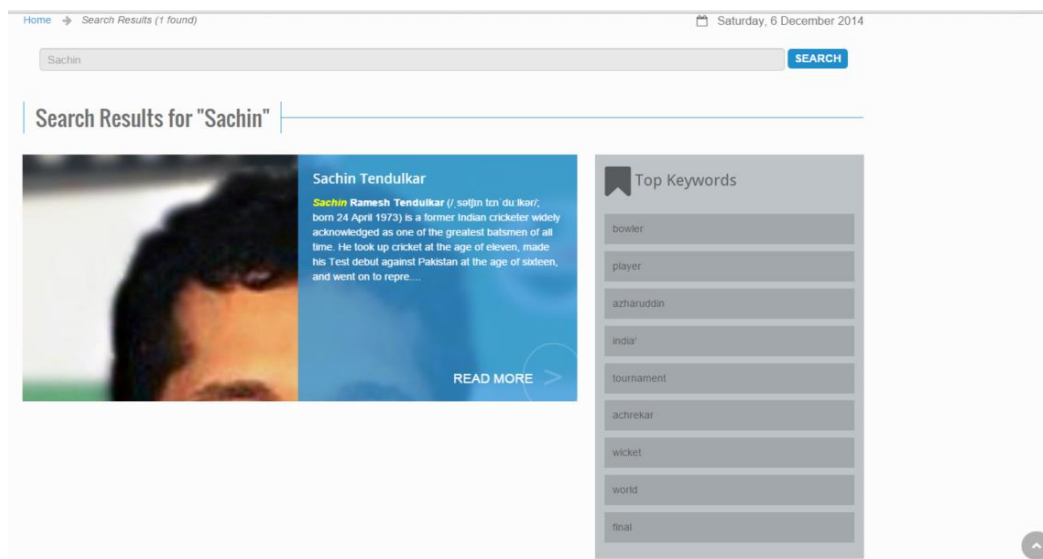


Figure 19 Dynamic Retrieval and Real-time Indexing



- Below is the main article page which is displayed after the user selects the document from the search results. This page contains the related tweets, related news articles, tags or important keywords related to the article, top trends or important terms from the news/twitter posts, relevancy score of the document with the query and the content of the Wikipedia article.



Home > Sachin Tendulkar

Saturday, 6 December 2014



Sachin Tendulkar

TAGS:

#Michael #Player #Andrew #Wanglal #Pw/project #Steve #Brian

#Donald #Graham #Turner #Alan #Gilbert #Salcolm #Simpson #England

#Shane #Davidson #Richi #Greens #Spilock #Chappel #Simon #Moran

#Richardson #Vicki

"Tendulkar" redirects here. For other people with the same surname, see Tendulkar (surname).

Sachin Tendulkar



Tendulkar at an awards event in January 2013

**Personal information**

Full name Sachin Ramesh Tendulkar

Born 24 April 1973

Bombay, Maharashtra, India

Nickname Tendy, Little Master, Master Blaster

Height 5 ft 5 in (165 cm)

Batting style Right-handed

Bowling style Right-arm medium, leg break, off break

Role Batsman

**International information**

National side India

Test debut (cap 15) 15 November 1989 v Pakistan

Last Test 14 November 2013 v Vijest Indies

ODI debut (cap 74) 18 December 1989 v Pakistan

**Our Relevancy Score**

for the query "Sachin Tendulkar"

100%

**Related Tweets**

> @shadex: WTH! Sachin gone.

> @santandregunathan: no sachin for WI tour!

> @vivekz @sachin\_cas i also feel the same

> @neerajaras. @roxin i was one of the guys out of 9.9L who called Sachin Obv. Sachin didn't pick up!

> @dehdreams: pathan & warne versus sachin. whom should i root for will stay loyal to my god

**Related News Articles**

**Indian Officials to Rule How 'Backward' Group Is**  
They burned buses during the morning rush hour. They threw stones at trains packed with passengers and at the police. By Monday evening, after a week ...  
[Read more >>](#)

**Indian Shepherds Stoop To Conquer Caste System**  
A fight for the right to be downwardly mobile exploded this week in north India, as a powerful community of Indian shepherds asserted that the best wa...  
[Read more >>](#)

**New Conflicts Accompany Nepal's Efforts at Democracy**  
A year after the return of democratic rule to Nepal, the scene in Parliament went something like this: No sooner had it officially opened for business ...  
[Read more >>](#)

**Hot Trends**

#gore

#sachin

#vict

#hose

Figure 20 Article Page