Crawl and Detect Duplicates on News Feeds

Jie Yu, Shiwen Cheng, Vagelis Hristidis

Department of Computer Science and Engineering

University of California at Riverside

jieyu@ee.ucr.edu

Abstract:

This project is part of the twitter time series project and the time series project is focus on study the problem of correlating social network (with focus on twitter) and RSS news activity data with real time events, and to do prediction of events based on the correlation, the prediction including stock market and real estate market, etc. This paper presents the work of crawling and detecting the duplicates on news feeds and put into the database for future use. Because of the large volume of RSS news feed source, the duplicates information can be large, the challenging part of my work is to parse the RSS source, detect the duplicates of the RSS feeds from different RSS source while keep updating the information from internet and then put data in category in database. SimHash algorithm is used for the detection of duplicates, and Cassandra is used as the storage database.

We present detailed experimental results including the statistics on the news we get per hour, during day or night and the percentage of the duplicates, we also showed some example of duplicate news, time experiments, graphs, etc. Our result shows that the method we used is relatively accurate to detect duplicates and the program is robust as a server program, the volume of the RSS feeds crawled can be used through predictor to predict the real time events.

Keywords:

RSS feeds, detect duplicates

1. Introduction

Scientists and researchers are trying to find ways to extract and understand the knowledge from the large volume data from internet, especially social network and RSS news, but at the same time, because of the different various sources from internet, there are a large volume duplicates from the web and how to crawl and pull data out of files, obsolete file formats, duplicate documents, and textual noise and at the same time keep store the data into database and keep up to data is a challenging issue. In this paper, we study the strategies for crawling, detecting duplicates and storing documents and keeping those documents up to date.

We start by carefully design the schema and database –Cassandra, then we study various algorithms for detect and removing duplicates, especially near-duplicates, simHash is proved to be the best algorithms to efficiently detect the near duplicates. Then the communication interface between the database and sources, and Cassandra database is selected because of its features of scalability and high availability without compromising performance, it is the perfect platform for mission critical data.

Our main contributions can be summarized as follows:

- We successfully pull data out of files, navigating through issues of character encodings, crawl the web and retrieving web pages, document feeds, and storing the documents using the Cassandra database system
- We use Simhash algorithms to detect the duplicates and remove the noise; the compared result showed on section xx and examples is given to show as the evident.

Roadmap: In section 2 we describe the background including a brief introduction to RSS and how it works, an example is given for explanation. In section 3, we described the program framework to give out a general idea of how the program works. The data structure is discussed in section 4 and the algorithm

used to detect duplicates is discussed in section 5 and in section 6 we also introduce the CLI command line interface and finally the result is discussed in section 7 and conclusion is presented in section 8.

2. Background

We started our presentation by background introduction to RSS, which is the data source of our project.

What is RSS? Rss is Really Simple Syndication is a family of web feed formats used to publish frequently updated works—such as blog entries, news headlines, audio, and video—in a standardized format. An RSS document (which is called a "feed", "web feed", [1] or "channel") includes full or summarized text, plus metadata such as publishing dates and authorship.

RSS is a dialect of XML. All RSS files must conform to the XML 1.0 specification, as published on the World Wide Web Consortium (W3C) website.

Features of RSS [2]

- RSS stands for Really Simple Syndication
- RSS allows you to syndicate your site content
- RSS defines an easy way to share and view headlines and content
- RSS files can be automatically updated
- RSS allows personalized views for different sites
- RSS is written in XML

Why RSS? RSS feeds benefit publishers by letting them syndicate content automatically. A standardized XML file format allows the information to be published once and viewed by many different

programs. They benefit readers who want to subscribe to timely updates from favorite websites or to aggregate feeds from many sites into one place. RSS was designed to show selected data.

Without RSS, users will have to check your site daily for new updates. This may be too time-consuming for many users. With an RSS feed (RSS is often called a News feed or RSS feed) they can check your site faster using an RSS aggregator (a site or program that gathers and sorts out RSS feeds).Since RSS data is small and fast-loading, it can easily be used with services like cell phones or PDA's Web-rings with similar information can easily share data on their web sites to make them better and more useful [2].

How to get RSS? RSS feeds can be read using software called an "RSS reader", "feed reader", or "aggregator", which can be web-based, desktop-based, or mobile-device-based. The user subscribes to a feed by entering into the reader the feed's URI or by clicking a feed icon in a web browser that initiates the subscription process.

What is RSS reader? The RSS reader checks the user's subscribed feeds regularly for new work, downloads any updates that it finds, and provides a user interface to monitor and read the feeds. RSS allows users to avoid manually inspecting all of the websites they are interested in, and instead subscribe to websites such that all new content is pushed onto their browsers when it becomes available.

Who should use RSS? Webmasters who seldom update their web sites do not need RSS and RSS is useful for web sites that are updated frequently, like:

- News sites - Lists news with title, date and descriptions
- Companies - Lists news and new products
- Calendars - Lists upcoming events and important days
- Site changes - Lists changed pages or new pages

An RSS Example [3]:

The processor (as the specification calls it) is often referred to colloquially as an XML parser.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="2.0">
<channel>
  <title>W3Schools Home Page</title>
  <link>http://www.w3schools.com</link>
  <description>Free web building tutorials</description>
  <item>
    <title>RSS Tutorial</title>
    <link>http://www.w3schools.com/rss</link>
    <description>New RSS tutorial on W3Schools</description>
  </item>
  <item>
    <title>XML Tutorial</title>
    <link>http://www.w3schools.com/xml</link>
    <description>New XML tutorial on W3Schools</description>
  </item>
</channel>

</rss>
```

3. Program Framework

Data acquisition and pre-processing: The RSS channel from the major big website including like yahoo, CNN, abc, US today, Google news, etc. The RSS usually contains some several typical channels like: world, US, business, technology, entertainment, sports, science, health, spotlight, education, polities, etc. some web has more detailed categories, in order to not missing any news may related to the time series events, we put all the channels as the sources. And put a tag on that indicating which channel it comes from. Below is an architectural figure showing all components, including Cassandra, and their interfaces. The frequency of checking RSS source is set to 1 minute and the frequency to reach the database is depend on the frequency of receiver of data, the program will write data into database immediately.

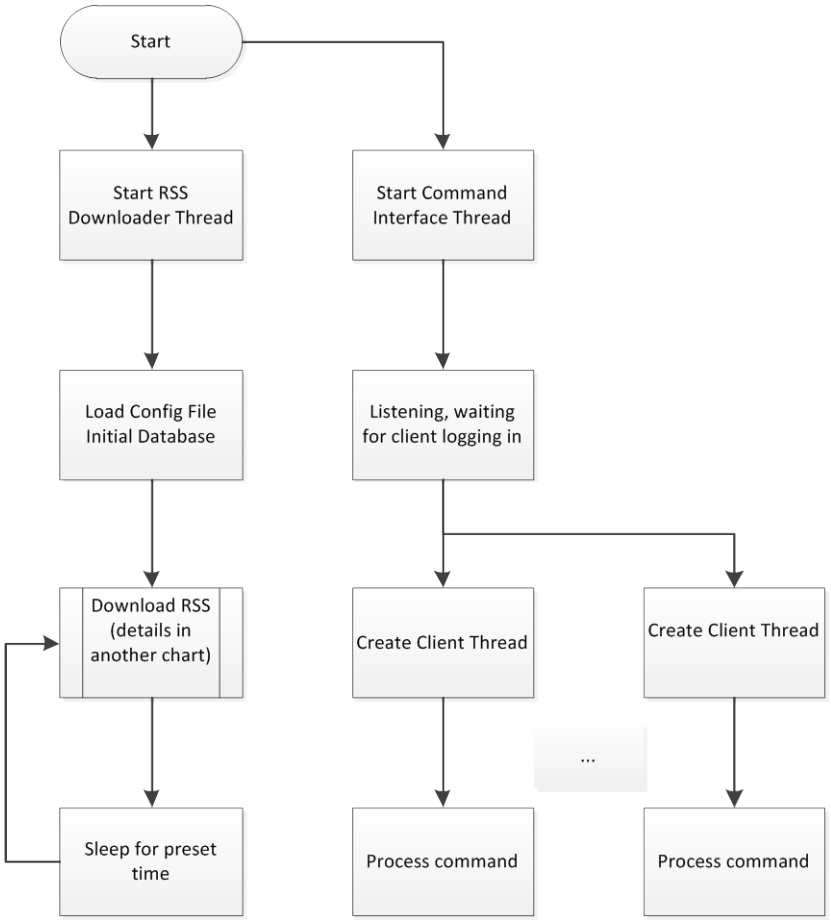


Figure 1. Program Flow

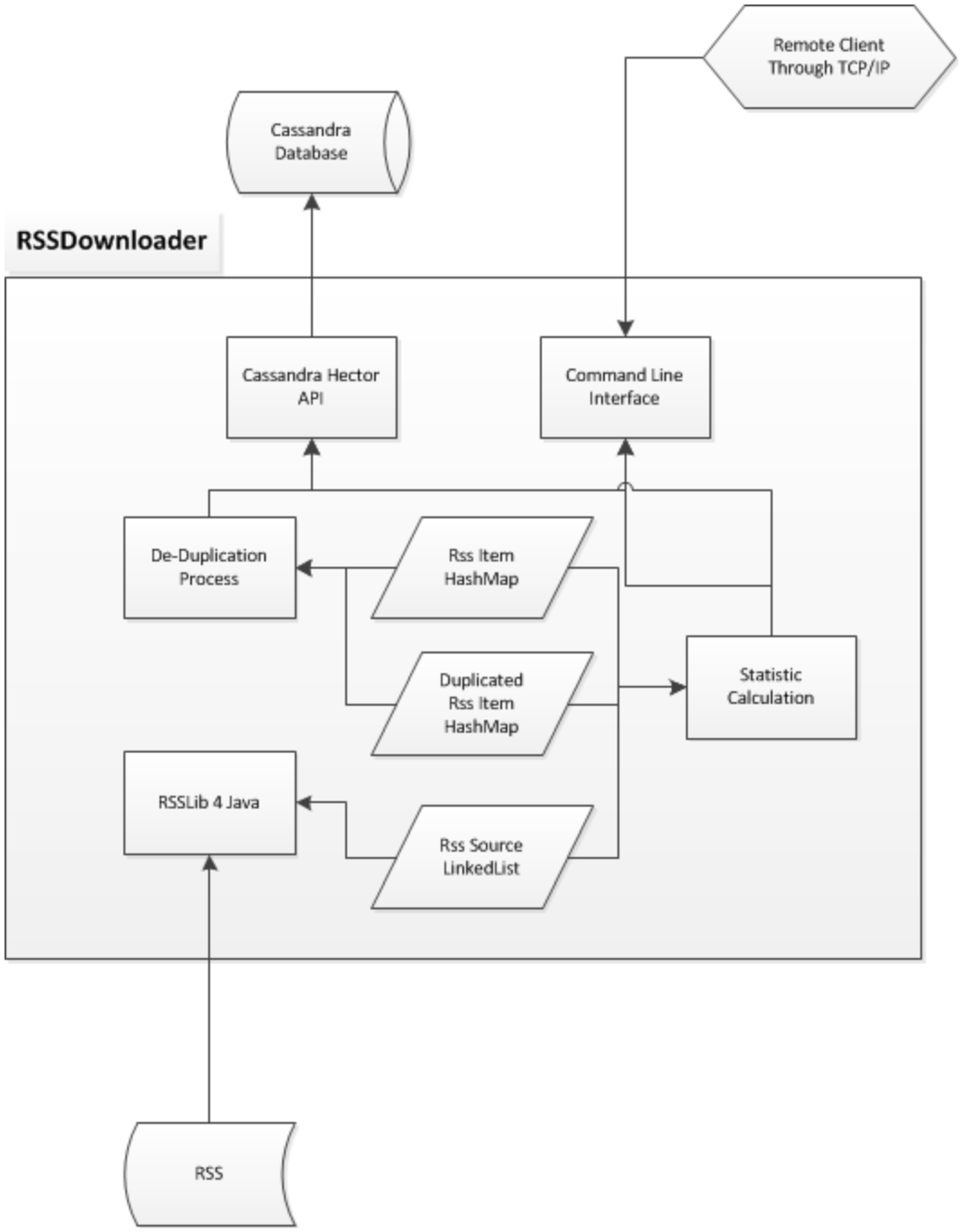


Figure 2. System Architecture

3.1 Java lib for crawl RSS

This RSS parser built for Java: RSS lib for Java (rsslib4j-0.2.jar) is used for crawling RSS.

```
// start a RSSHandler
RSSHandler remoteRSSHandler = new RSSHandler () ;
// Parse RSS data, store crawled data into remoteRSSHandler
RSSParser. parseXmlFile(new URL("RSS address"), remoteRSSHandler,false);
// get RSS channel
RSSChannel channel =remoteRSSHandler.getRSSChannel ();
```

3.2 How the crawling program works

3.2.1 Function description

This is a server program. The function of this program including connect Cassandra database using config file (rss.cfg) and then crawl RSS data from webpage at certain time period. The Rss data was written into database after removing the duplicate.

Another function of this server program is network based command line interface, CLI. It can be connected with a client and then real time monitors the status of the server program and manages the server.

3.2.2 Program framework

This program has two threads, one thread is used for data crawling, and the other is work as the network based command line interface, CLI. RssDownloader.Downloader TimerTask.java is the crawling data thread and the RssDownloader.Utility.CmdServer.java is the command line interface, CLI.

4 Data Structure

4.1 Main data structure

There are two main date structures in this project: Class RssItem and Class RssSource. Both of them are defined in standalone java file. They have everything needed for RSS Source and RSS Items.
An instance of RssSource is created in RSS Download thread. This is a linkedlist for RssSource. (`RssSourceList rssSource`). After it is created all Rss Source stored in the database will be loaded. And then Rss Download thread will go through each RSS source.
Two instances of RssItem are created in RssItemList. Both of them are based on HashMap. One is used for storing normal RSS Item, the other one is used for duplicated RssItem.

```
public class RssSource
    private String name;          // the unique name for this Rss Source
    private String url;           // the RSS URL for this Rss Source
    private String category;      // the category this RSS belong to
    private String lastBuildDate; // the last build date for one release
    private String titlePrefix;   // the RSS Items in some RSS source has fixed
prefix. we can use this parameter to remove prefix when downloading
     private long rssItemNum;     // how many RSS Items have been downloaded from
                                     this source
```

```
public class RssItem
    private String title; // the title for this RSS Item
    private String description; // the brief content for this RSS Item
    private String author; // the anthor for this RSS Item
    private String url; // the URL for details of this RSS Item
    private Date pubDate; // the Publish date for this RSS Item
    private Date downloadDate; // the downloaded Date for this RSS Item
    private long simHash; // The simHash Value for this RSS Item. It is calculated
based on RSS Item title and description
    private String rssSource; // the name of its RSS Source, it can be seen as
foreign key to RSS Source
    private long distanceSum; // If this is a primary RSS Item which has many
duplicated RSS Items, this is sum of the total hamming distance between each
duplicated RSS Item. otherwise it is no meaning
```

Figure 2. Schema and description

4.1.1 RssSource Package

There are two classes in this package, one is RssSource.java, this class works as describing single Rss source, the other is RssSourceList.java, and this is used to store the RssSource list. When the program start, RssSourceList.java will get data from Cassandra and then read RSS source to the object class.

In rss.schema, the column family of Rss source including a three columns: url- the url for this rss source, category- the category for this RSS source, and the titlePrefix – the prefix for the title, in some RSS Source, there are some prefix at the head of RSS title, it is not good for similar RSS item distinguishing, the prefix will be removed when processing it (RssDownloader)

4.1.2 RssItem Package

The package also including two classes, one is RssItem.java, this class is used for describing the single RssItem, another class is RssItemList.java. This is used for operate and storage Rss Item downloaded from webpage, this program keep every RSS Item downloaded from webpage in Cassandra database in real time. Because of the large amount of duplicated news feed from various RSS news webpages, we need to temperately storage RSS Items in from N (N can be defined by developer) days in database to detect the duplicate news and remove them. So there are two important data structures in this class, rssHashTable which is used temperately storage RssItems downloaded from webpage, the data type of RssHashTable is Hashmap, the key is the SimHash value of Rss Item, and the value is the object of every RSS Item. Another rssDuplicated is used for storage the detected duplicate RssItem. The data type of rssDuplicated is also HashMap, the key is the SimHash value of major RSSItem among the pool of the duplicated RSS Items and the value is the linked list for storage the duplicated RSSItems.

In Rss.schema, the column family for RSS Item is a super column family, the key for this super column family is the publish date of RSS Items, so the RSS items with same publish date will be stored together and the subkey will be the sim-hash value of RSS items which can be seen as unique ID for this RSS items. There are seven columns including: title, author, brief, content, source, url and pubdate. The column family for duplicated RSS item is also a super column family, the key for this super column family is the sim-hash value of primary RSS item which is stored in the RSSItem, so the similar RSS items are stored together, and the key of this super column is the sim-hash value for duplicated RSS Item. In the column family there are eight columns including: title, author, brief,content source, url, distance and pubdate.

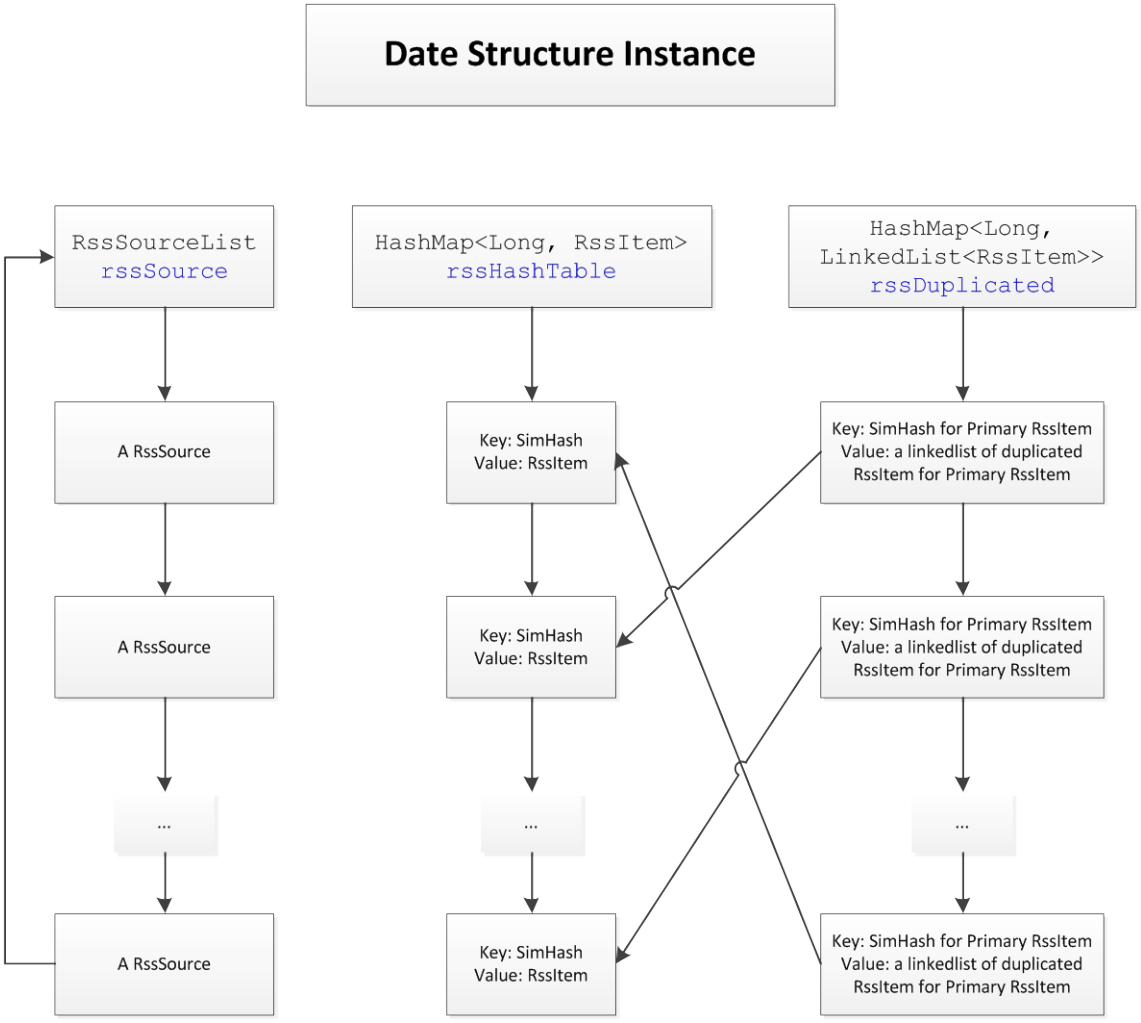


Figure 3. Data Structure instance

4.2 Data flow

4.2.1 Major data flow

The steps from download RSS data from webpage to storage the RSS into database including: download; detect duplicates, storage into database. The following pseudo-code will describe this procedure, this part of code is in DownloaderTimerTask.eachRun(),

```
While (there is still unprocessed RSS source in the RSS source list)
{
  Get RSS source;
  If (the pubdate of RSS source is newer than last time)
   {
      While (there is still unprocessed RSS Item in the RSSitem list)
      {
       Calculate RSSItem SimHash value
       Build a new RssItem object
       ret = put new RSS item in the rss item list
       if (ret ok) // successfully put RSS into RSS Item list
           {
            Update the database
            }
      }
   }
}
```

This is main process for this project, it is defined in function (`public void eachRun()`) in (`class DownloaderTimerTask`) ,

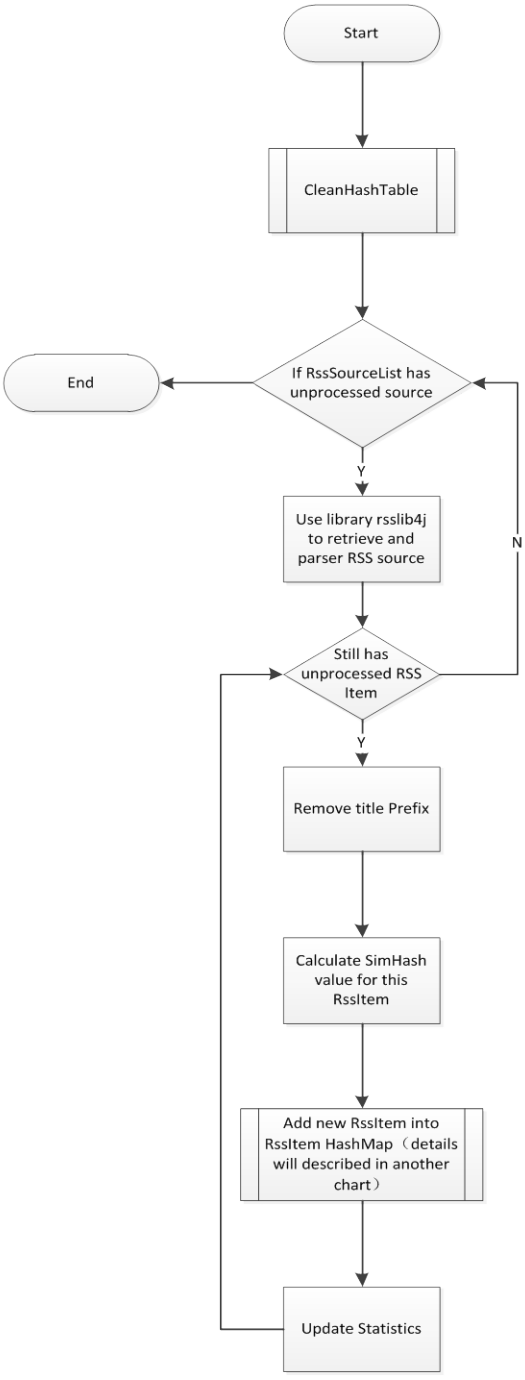


Figure 4. RSS Downloading processing

Add New RssItem Processing

Adding new RssItem is described in the function(`public ItemAddReturn addRssItem(RssItem rssItem)`) in (`public class RssItemList`). Firstly it will go through de-duplicate process and then the new RSS Item will be stored in to database in the proper table.

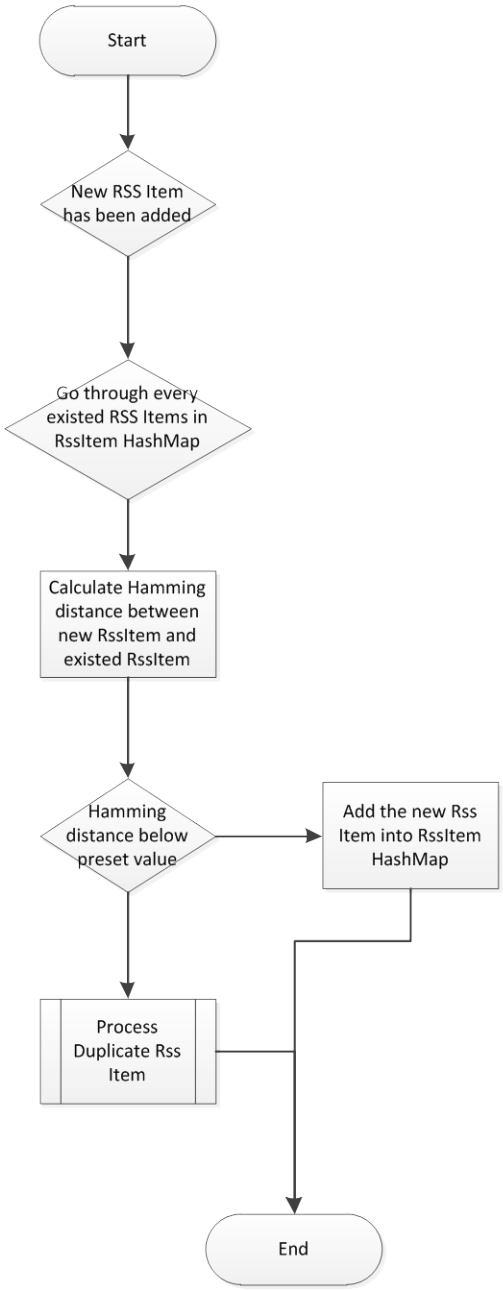


Figure 5. Add new RSSItem processing

4.2.2 How to remove duplicate

Because of the features of RSS, there will exist a lot identical or similar RSS items from different RSS, the key method for detect and remove duplicates is by comparing the simHash value for two RSS items. If the hamming distance of the simhash is less than the set value, then the two RSS Items can be considered as similar RSS Item. Then after get a new RSS Item, we need to go through the whole rssHashTable, calculate the hamming distance of new RSS item and the rss item in the rssHashTable. This is the most time consuming and most complicated part of this program, and with the increase of rssHashTable, this process will become slower and slower. This part code can be more efficient in the future. There is a need for simhash to do more effective search and comparison. SimHash is a vector operator, if we can do vector ordering, and then the time consumed by search can be efficiently reduced. This part of work can be discussed in details in the following part “How to remove duplicate”

Here is the pseudo code:

```
If (new RssItem is not in the rssHashTable)
{
   While (traverse rssHashTable)
   {
     If (compute HammingDistance(newRssItem, eachRssItem < pre-set value)
        {
          newRssItem is the duplicated RSS Item, process RSS duplicates
        }
   }
}
```

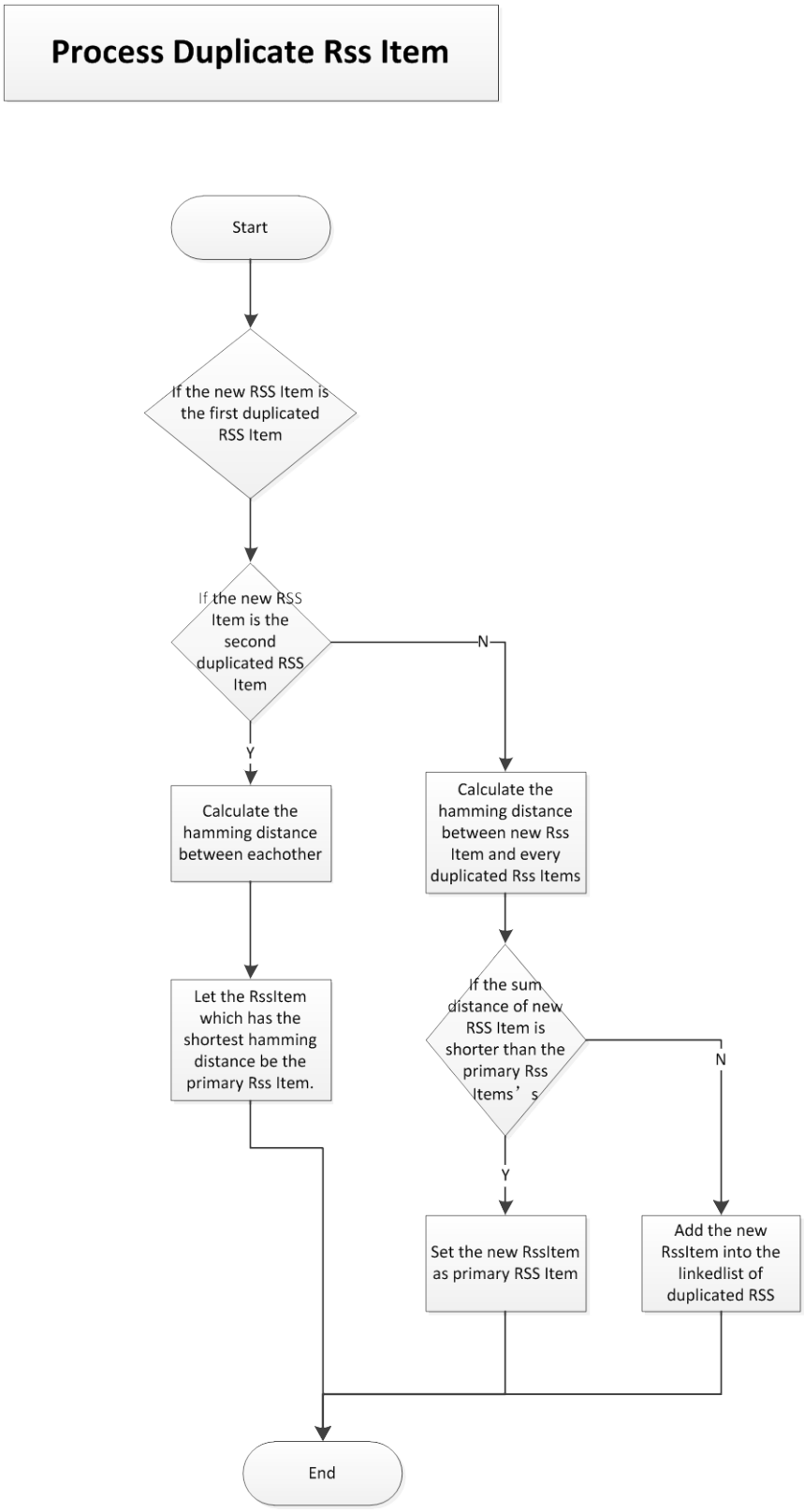


Figure 6. Process duplicate RssItem

Clean HashTable

The RSS Items stay in the memory (RssItem HashMap) is just for de-duplication. While the RSS news is time limited. So old date(RSS Items) has little use for de-duplication.
In this project, there is a value to set the RSS Item live period. The RSS Items which is old than this time, will be removed from memory. This function is processed by function: (`public void cleanHashTable()`)

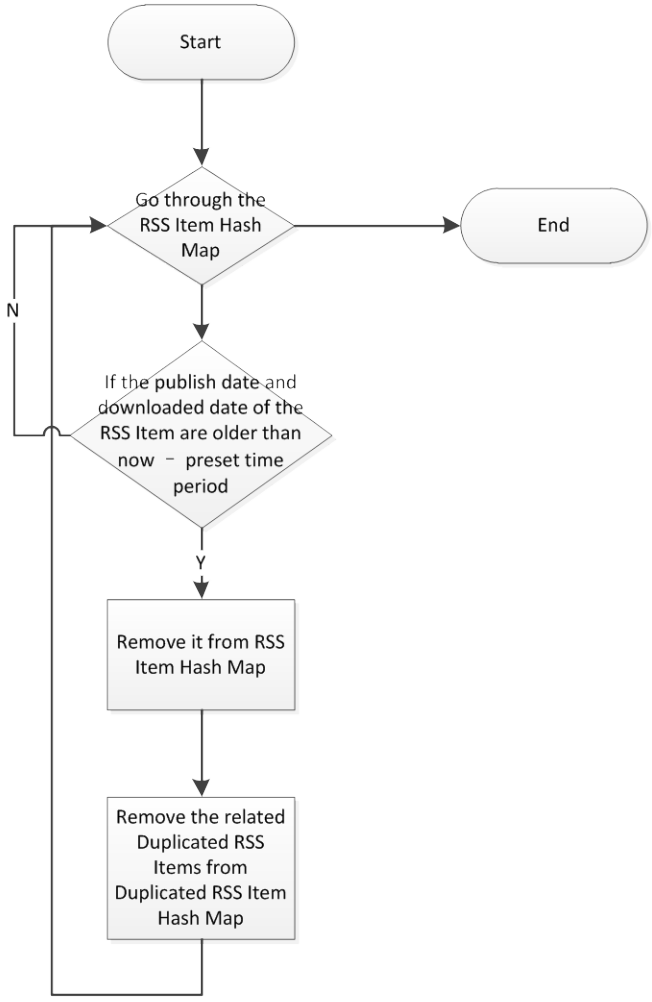
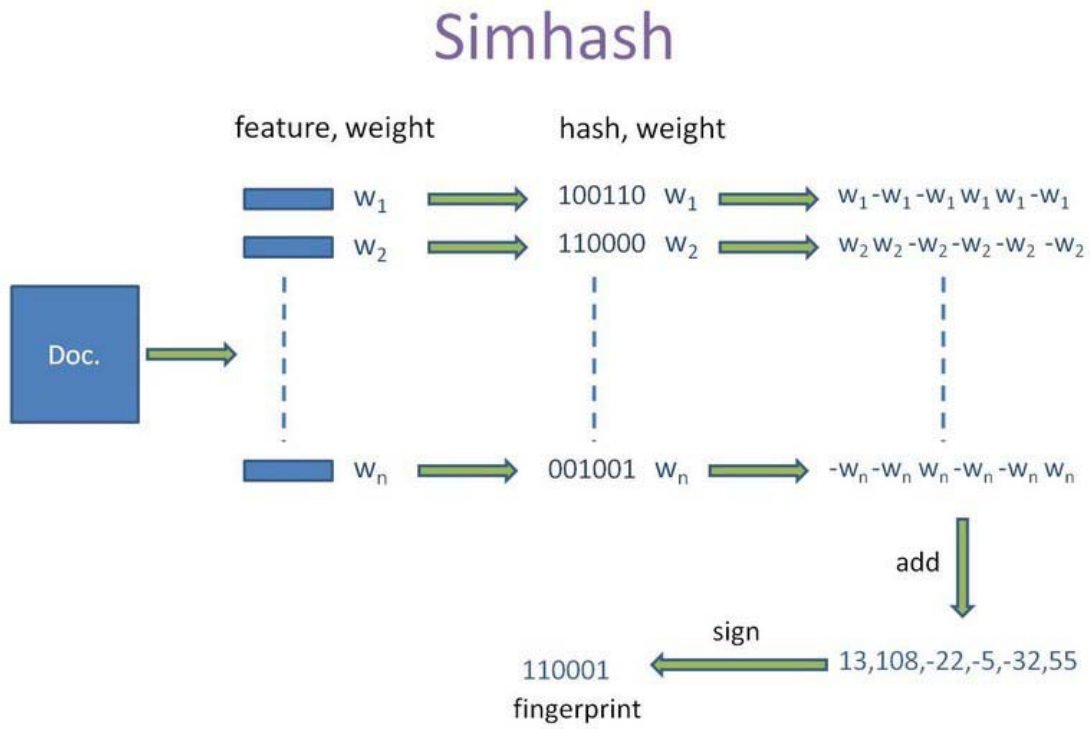


Figure 7 . Clean Hashtable

5 Remove duplicates algorithm

5.1 SimHash algorithm

Simhash was developed by Moses Charikar and is described in his paper [4] simhash is useful because if the simhash bitwise hamming distance of two phrases is low then their jaccard coefficient is high



5.2 The Implementation of SimHash algorithm in this program

In this program, we used simHash algorithm and calculate the simhash by the following function:

Public static longcomputerOptimizedSimHashForBytes(byte[]data, int offset, int length);

Then for remove the rss duplicates part, the following function is used for the calculation of hamming distance: public static int hammingDistance(longhash1,longhash2);

5.2.1 What data can be used to calculate SimHash

The calculation of SimHash is based on the calculation of strings. So the length of the string will directly decide the time for SimHash to calculate. If we choose title as the seed to calculate simhash, then the time consuming will be lower but the error will happen as below:

Problems:

Title: 10 things you need to know today: November 16, 2012

Title: 10 things you need to know today: November 17, 2012

Title: 10 things you need to know today: November 18, 2012

We can see that these RSS Items are different RSS Items, but the hamming distances between each other are very short. So we will choose “Title + Brief description” as the seed for SimHash.

5.2.2 Result Analysis

In the duplicates algorithm, choose the threshold for similar RSS Item hamming distance is very important, if the threshold is too low, there will be duplicate RSS missing, but if the threshold too high, the non similar RSS will be considered as the duplicated ones.

Below is some examples of the hamming distance of the string:

String1	String2	Hamming Distance
List of National League MVP award winners	List of National League MVP award winners	6
Golf-McIlroy set to miss cut in Hong Kong after putting problems	McIlroy set to miss cut in Hong Kong after putting problems	3
Battery maker A123 files for bankruptcy protection	Alien Life May Require Rare 'Just-Right' Asteroid Belts	18
Explosion on bus in Kenya capital kills at least 5	Riot in Kenya after bus bombing kills 7 in capital	14
Charles Johnson criticizes unnamed Panthers teammates on Twitter after Bucs loss	An ‘Encore’ Life Beckons … on the Far Side of Midlife	19

we can see that when hamming distance is above 14, the result will be very bad, so the best value for hamming distance would be 8-12.

5.2.3 The volume of crawled data

Every RSS source publish different numbers of RSS news every day, on average is about 10-25 items, and update 3 times per day, if set 100 RSS Source, then the total number of RSS news can get would be 100*18*3= 4800, the detailed statistic result refer to section 8

6. Database

6.1 Introduction to Cassandra

Apache Cassandra is an open source distributed database management system. It is a NoSQL solution initially developed by Facebook, the developer Jeff Hammerbacher, who led the Facebook Data team describe Cassandra as a big Table data model running on an Amazon Dynamo-like infrastructure. The main features including: decentralized, supports replication and multi data center replication, scalability, fault-tolerant, tunable consistency, mapReduce support, query language [5].

6.1.1 Data Model [6]

Every row is identified by a unique key. The key is a string and there is no limit on its size.An instance of Cassandra has one table which is made up of one or more column families as defined by the user.The number of column families and the name of each of the above must be fixed at the time the cluster is started. There is no limitation the number of column families but it is expected that there would be a few of these. Each column family can contain one of two structures: supercolumns or columns. Both of these are dynamically created and there is no limit on the number of these that can be stored in a column family.

Columns are constructs that have a name, a value and a user-defined timestamp associated with them. The number of columns that can be contained in a column family is very large. Columns could

be of variable number per key. For instance key K1 could have 1024 columns/super columns while key K2 could have 64 columns/super columns. “Supercolumns” are a construct that have a name, and an infinite number of columns assosciated with them. The number of “Supercolumns” associated with any column family could be infinite and of a variable number per key. They exhibit the same characteristics as columns.

6.2 Cassandra schema

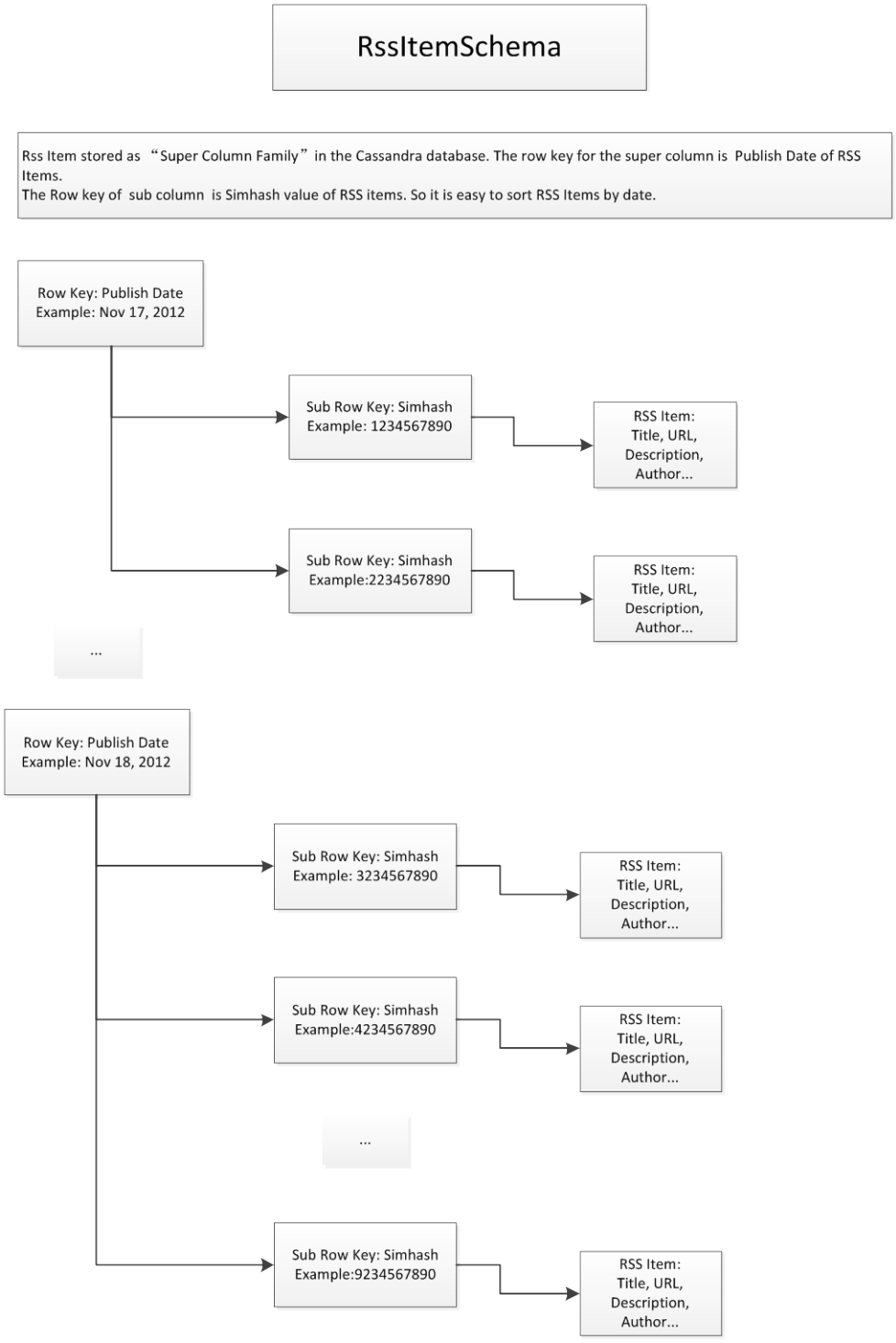


Figure 8. RSS Item schema

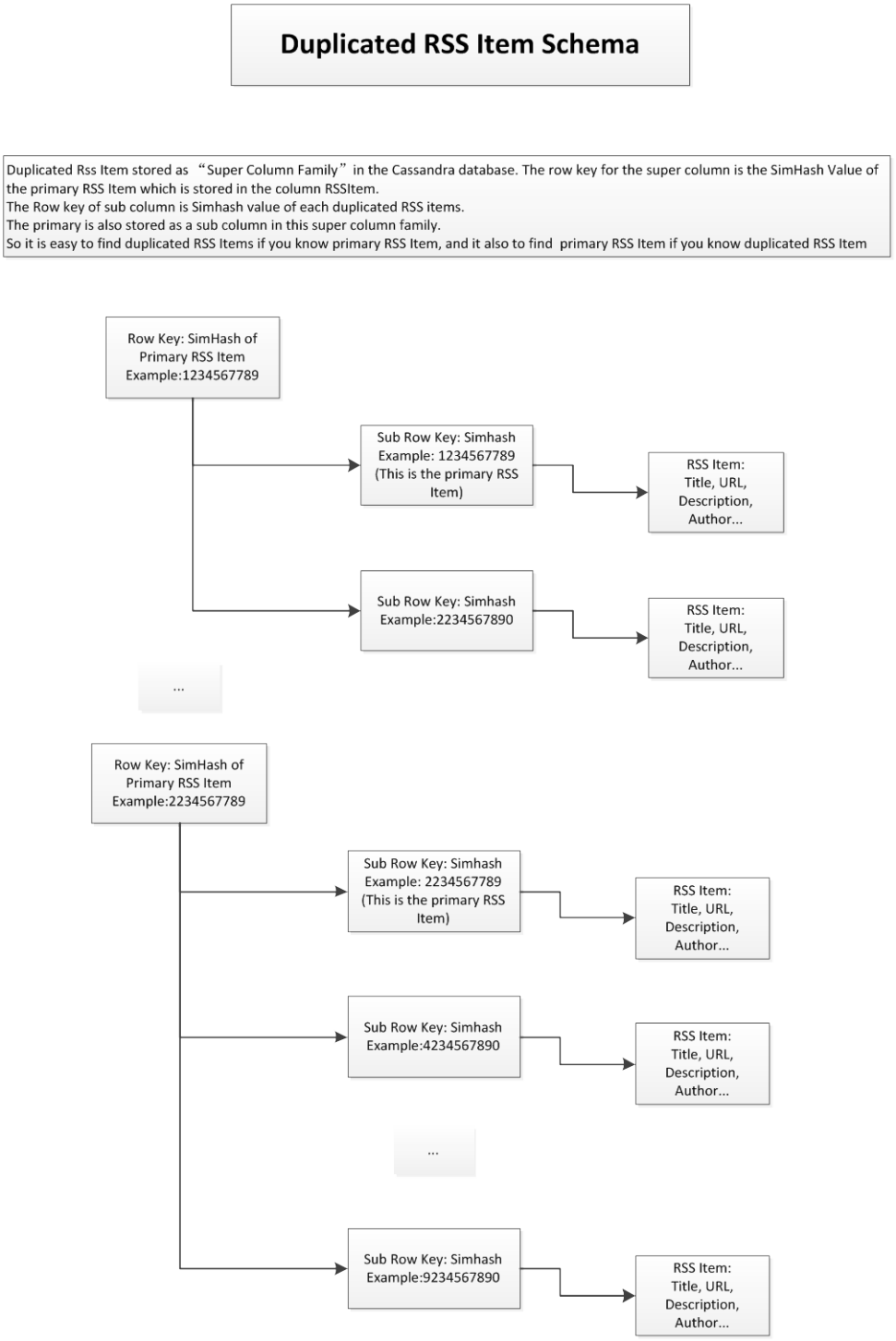


Figure 9. Duplicated RSS Item schema

7: Command line interface

The command line interface is a typical server/client TCP/IP model; it is independent in one thread, the server connect client through the function Accept, when the client initiate the request to server, a new thread is built for the server operation. Below is the pseudo code:

```
While (1) {
        clientSocket= accept(serverSocket);
        Thread clientThread = newThread(clientSocket);
        ClientThread start()
      }
```

In every client thread, there is a command line interface, in charge of process every command, below is the pseudo code:

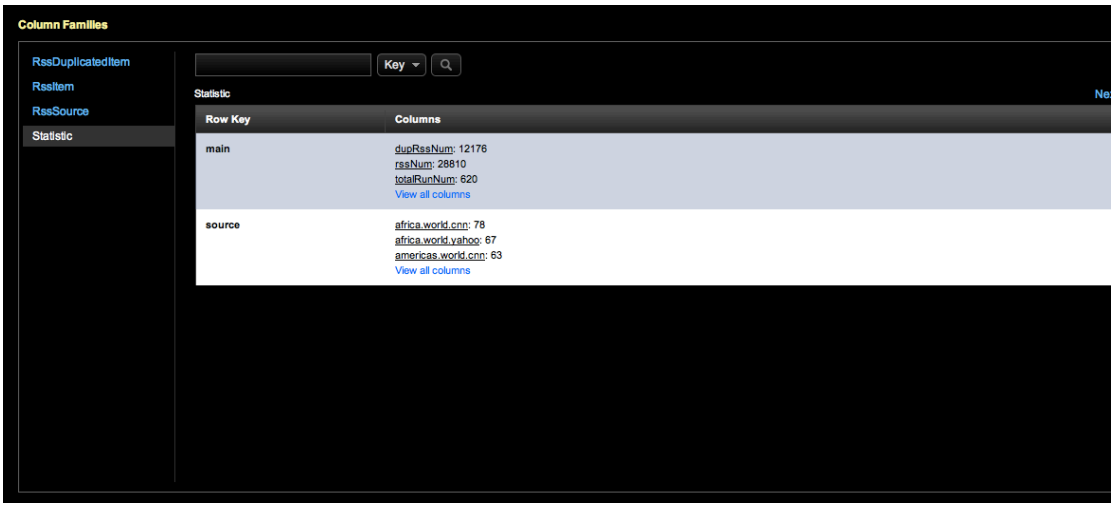
```
 While (inputCmd=inputIO.getline()!=null)
{
Foreach(cmd:commandList) // transverse every command in command list
{   If (inputCmd.compare(cmd.cmdstr)) // if the input cmd is consist with one of the command
    {
      cmd.exe(); //execute the command
     }
  }
```

}

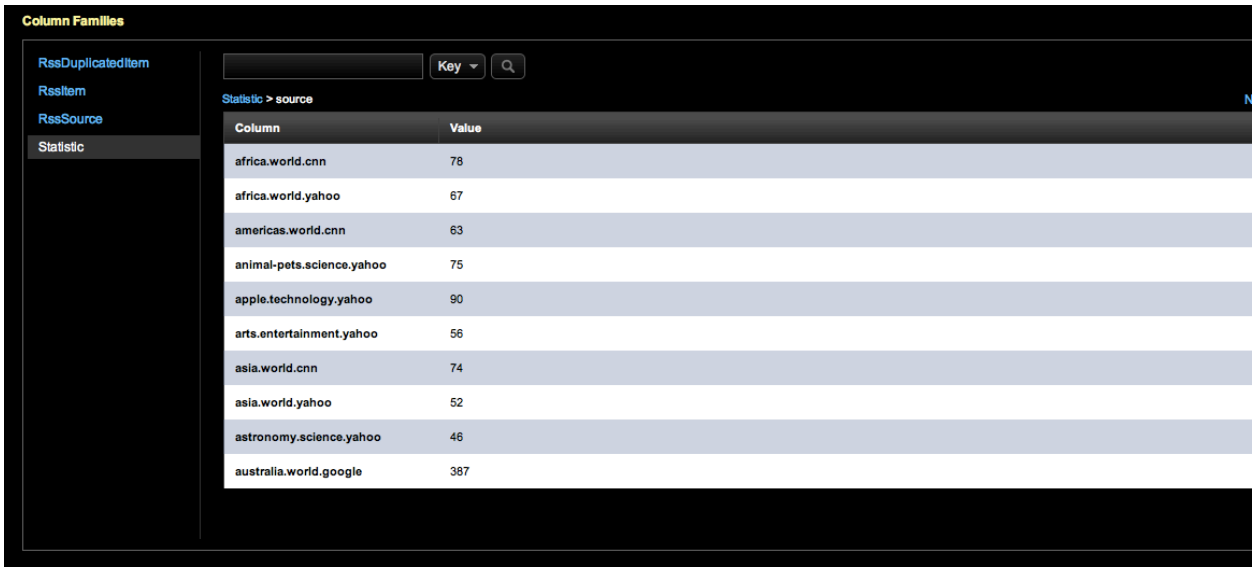
8. Result and Statistic analysis

We set up the experiments keep running the program to get some statistic result.

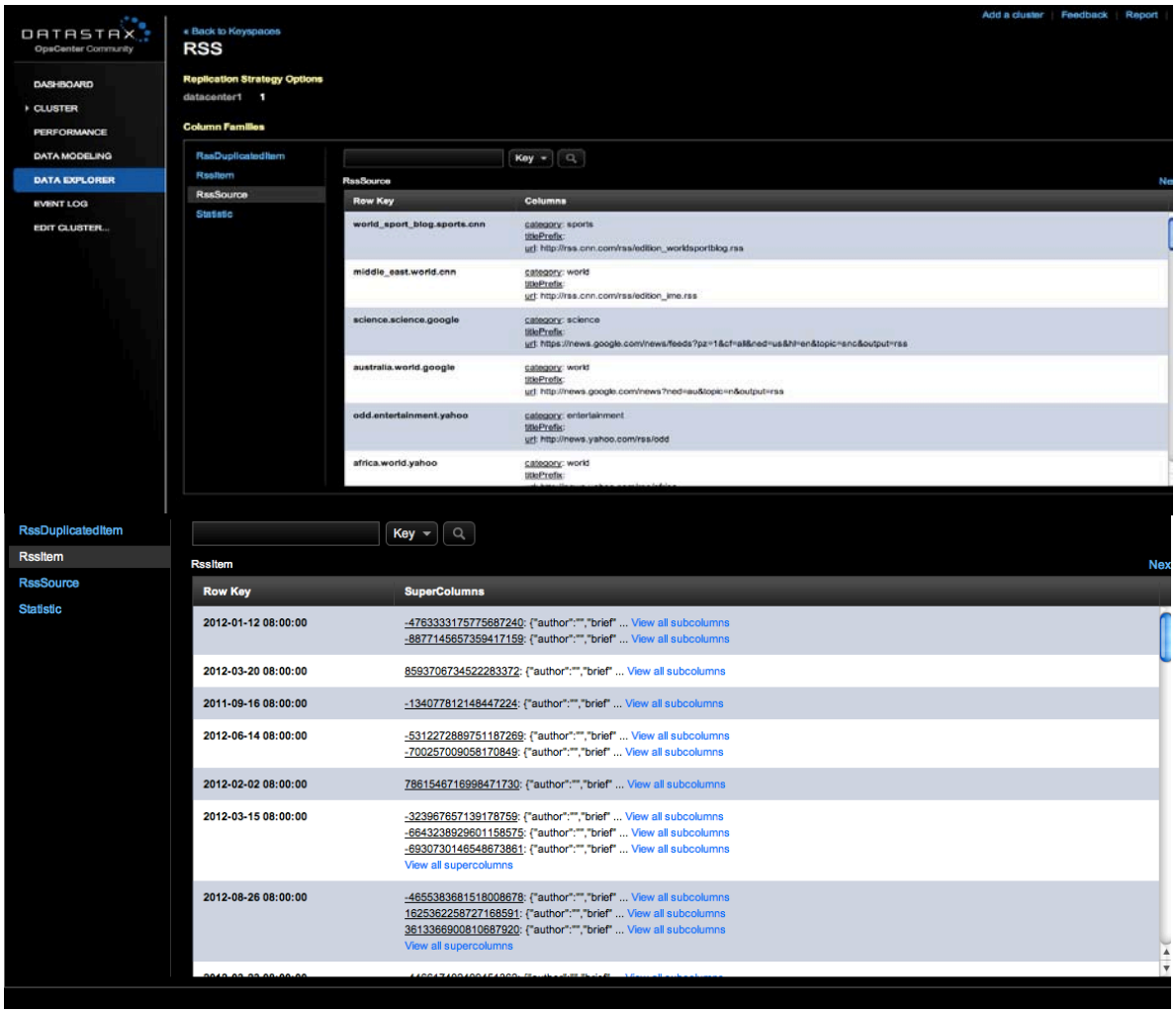
Parameters and Result: We keep the program running for 1069 minutes, about 18 hours, and the program run 620 times. The result showed the total collected RSS number is 28810 and the duplicate RSS number is 12176, so the percentage is about 42% duplicates. So the news we get on average is about 1600/ hour and out of them is 676/ hour.



The total RSS feeds sources are about 160. And some source returns more news feed has high value as below:



Here are the examples of RSS Item and RSS source:



Here is an examples of duplicate news, from different sources：

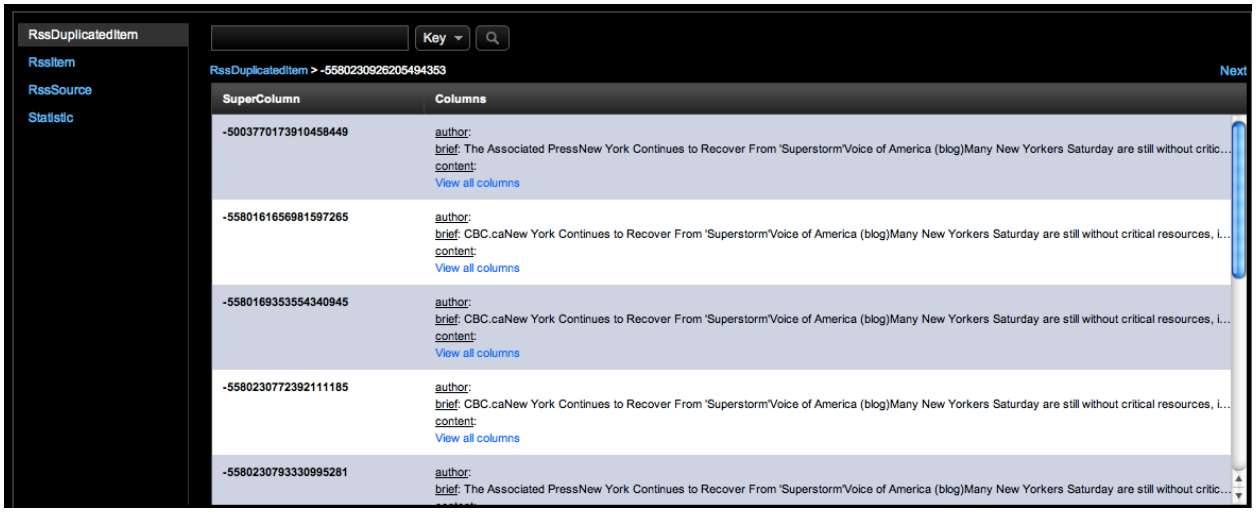
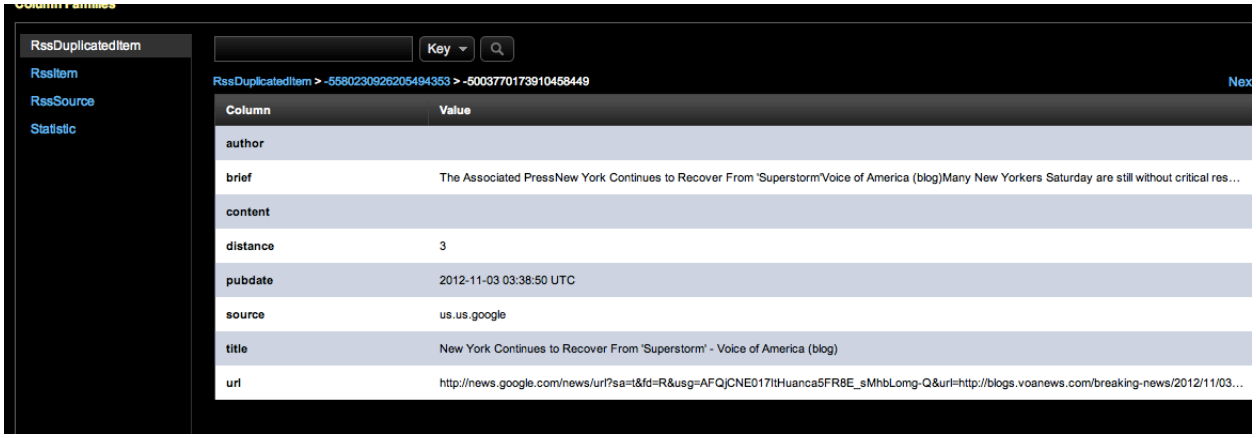
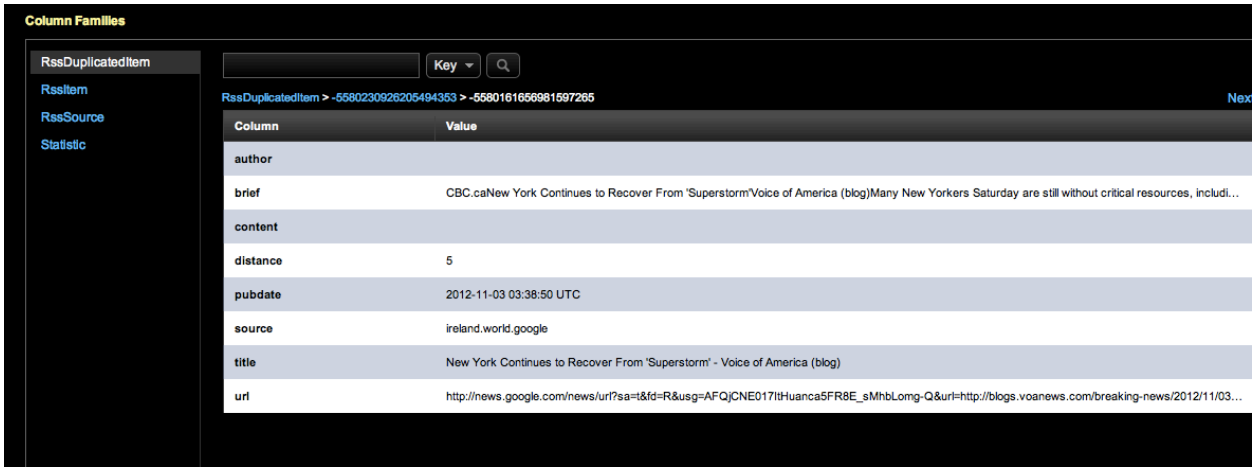


Figure above show all the duplicates under one super column and below figures show the details of each one duplicated item, the result prove that the algorithm can efficiently detect duplicates and put together.





9. Conclusion

The command line interface is a typical server We presented a framework of extract data from RSS news feeds and put the data into database for future use for the predictor to find a correlation between RSS news and time series event. We used SimHash algorithm to detect the duplicates and the experiment result shows xx numbers of RSS feeds we get per day and the duplicates contains xx %. These data is useful as a part of the source of a predictor to study the real time events such as trading strategy in the stock market.

10. Reference

[1]Wikipedia

[2] http://www.w3schools.com/rss/default.asp

[3] http://www.w3schools.com/schema/schema_example.asp

[4] http://www.cs.princeton.edu/courses/archive/spring04/cos598B/bib/CharikarEstim.pdf

[5] http://en.wikipedia.org/wiki/Apache_Cassandra

[6] http://www.facebook.com/note.php?note_id=24413138919

11. Appendix

RSS Schema code:

```
@brief create Keyspace for RSS Downloader
create keyspace RSS
with placement_strategy = 'NetworkTopologyStrategy'
and strategy_options = {datacenter1 : 1}
and durable_writes = true;
use RSS;
* @brief   the column family for Statistic, the items in this column
 *          family are dynamic. it will be rely on which row it will be
 * @key     the name of the statistic, such as: main, source, error
 *          main
 *              totalRunTime
 *              totalRunNum
 *              rssNum
 *              dupRssNum
 *          source, the statistic for each rss source
 *              [nameOfRssSource]
 *          error
 *              [errors, name field is the content of the error,
 *               value is left blank]
 *
 */
```

```
create column family Statistic
    with column_type = 'Standard'
    and comparator = 'UTF8Type'
    and default_validation_class = 'LongType'
    and key_validation_class = 'UTF8Type';


/**
 * @brief   the column family for Rss Source
 * @key     the name for this RSS Source, it is a UTF8 value
 *
 * @column url          the url for this RSS Source
 * @column category     the category for this RSS Source
 * @column titlePrefix  the prefix for the title, in some Rss Source, there are some prefix at the head of RSS title, it is not good for similar RSS Item distigiushing. The RssDownloader can remove prefix when processing it.
 */
create column family RssSource
    with column_type = 'Standard'
    and comparator = 'UTF8Type'
    and default_validation_class = 'UTF8Type'
    and key_validation_class = 'UTF8Type'
    and column_metadata = [
        {column_name: url, validation_class: UTF8Type}
        {column_name: category, validation_class: UTF8Type}
        {column_name: titlePrefix, validation_class: UTF8Type}
    ];
```

```
/**
 * @brief    the column family for Rss Item. it is a super column family. the key for this super column family is Publish Date of Rss Items. So Rss Items with same publish date will be stored together.
 * @key     publish date of Rss Items
 * @subKey  the sim-hash value of Rss Items which can be seen as unique ID for this RSS Items
 *
 * @column title        RSS Title
 * @column author       the author of this RSS item
 * @column brief        brief description of RSS item
 */
create column family RssItem
   with column_type = 'Super'
   and comparator = 'UTF8Type'
   and subcomparator = 'UTF8Type'
   and default_validation_class = 'UTF8Type'
   and key_validation_class = 'UTF8Type'
   and column_metadata = [
      {column_name: title, validation_class: UTF8Type}
      {column_name: author, validation_class: UTF8Type}
      {column_name: brief, validation_class: UTF8Type}
      {column_name: content, validation_class: UTF8Type}
      {column_name: source, validation_class: UTF8Type}
      {column_name: url, validation_class: UTF8Type}
      {column_name: pubdate, validation_class: DateType}
```

```
    ];


/**
 * @brief the column family for duplicated RSS Item, it is a super column family. the key for this super column family is the sim-hash value of primary RSS Item(which is stored in the RSSItem). So the similar RSS Items are stored together.
 * @key   the sim-hash value for duplicated RSS Item
 *
 */
create column family RssDuplicatedItem
    with column_type = 'Super'
    and comparator = 'UTF8Type'
    and subcomparator = 'UTF8Type'
    and default_validation_class = 'UTF8Type'
    and key_validation_class = 'UTF8Type'
    and column_metadata = [
        {column_name: title, validation_class: UTF8Type}
        {column_name: author, validation_class: UTF8Type}
        {column_name: brief, validation_class: UTF8Type}
        {column_name: content, validation_class: UTF8Type}
        {column_name: source, validation_class: UTF8Type}
        {column_name: url, validation_class: UTF8Type}
        {column_name: distance, validation_class: UTF8Type}
        {column_name: pubdate, validation_class: DateType}
    ];
```

```
/**
 * set the initial RSS Source,
 *
 * move it into another file
 *
 * Set RssSource['cnn.com']['url']='http://rss.cnn.com/rss/edition_business360.rss';
 * Set RssSource['cnn.com']['category']='General';
 * Set RssSource['cnn.com']['titlePrefix']='';
 *
 *      Set      RssSource['usatoday.com']['url']='http://rssfeeds.usatoday.com/UsatodaycomMoney-
Healey';
 * Set RssSource['usatoday.com']['category']='General';
 * Set RssSource['usatoday.com']['titlePrefix']='Test Drive:';
 *
 * \
Set RssSource['marketwatch.com']['url']='http://feeds.marketwatch.com/marketwatch/financial/';
 * Set RssSource['marketwatch.com']['category']='General';
 * Set RssSource['marketwatch.com']['titlePrefix']='';
 */
```