# Information Search on the Internet Practical Project

# *Twitter Search Aggregation based on streaming directly to browser with Node.js and HTML5 websockets*

Josef Dabernig

Institute of Software Technology and Interactive Systems
Vienna University of Technology
Favoritenstraße 9-11/188-1. 1040 Vienna. Austria/Europe

Dieter Merkl dieter.merkl@ec.tuwien.ac.at

**Abstract.** *Twitter Search Aggregation* prototypes a light-weight approach to aggregating data optained from twitter via search. Using modern web technologies as a *node.js* server and a JavaScript-driven *HTML5 websockets client*, *Twitter Search Aggregation* is an exemplary application of such post-processing based on twitter. It allows the end-user to *observe* a given topic by executing *real-time text search* and *live-tracking* of *twitter entities* such as *hashtags* and *user mentions*.

**Keywords: twitter, search, node.js, html5, websockets, web2.0, aggregation**

## Introduction

*Twitter* nowadays enables real-time communication of short-text messages and is used by many people all over the world. Users share updates on different *topics* with their *followers*. The emerging popularity of this medium also led to a vast variety of *mash-ups* and *services* building solutions on top of the *data* generated within twitter.

*Twitter* stands among the “ten-most-visited websites by Alexa's web traffic analysis”[1]. Yet, many twitter analytics tools exist, some of the most popular are listed at . As twitter is a microblogging platform, communication works via sending short-text messages limited to 140 characters. “*Users can group posts together by topic or type by use of hashtags – words or phrases prefixed with a "#" sign.[41] Similarly, the "@" sign followed by a username is used for mentioning or replying to other users*”[2]

*Twitter Search Aggregation* is an exemplary application of such post-processing based on twitter. It allows the end-user to *observe* a given topic by executing a *real-time text search* using the *Twitter Search API*. All observed tweets will then be presented to the user as a typical twitter stream with the addition, that *aggregation* happens. *Twitter Search Aggregation* introduces a *live-tracking* feature of *twitter entities* such as *hashtags* and *user mentions*. The appearance of any such entity will be recorded and presented to the user in order to show him the most relevant keywords (*hashtags*) or user references (*user mentions*) for the search that is currently performed, in real-time.

Using modern web technologies as a *node.js* server and a JavaScript-driven *HTML5 websockets client*, *Twitter Search Aggregation* prototypes a light-weight approach to aggregating data optained from twitter via search. The prototype builds upon the existing *twitter-nodejs-websocket* project (*Twitter streaming directly to browser with Node.js and HTML5 websockets*) which originally has been developed by Andre Goncalves and published at github:
https://github.com/andregoncalves/twitter-nodejs-websocket

*Twitter Search Aggregation* is neither intended to be a fully-functional nor thoroughly-tested program, but instead intends to illustrate the basic idea of aggregating and summarizing twitter entities by providing an interactive demo application.

[1]http://en.wikipedia.org/wiki/Twitter#Rankings

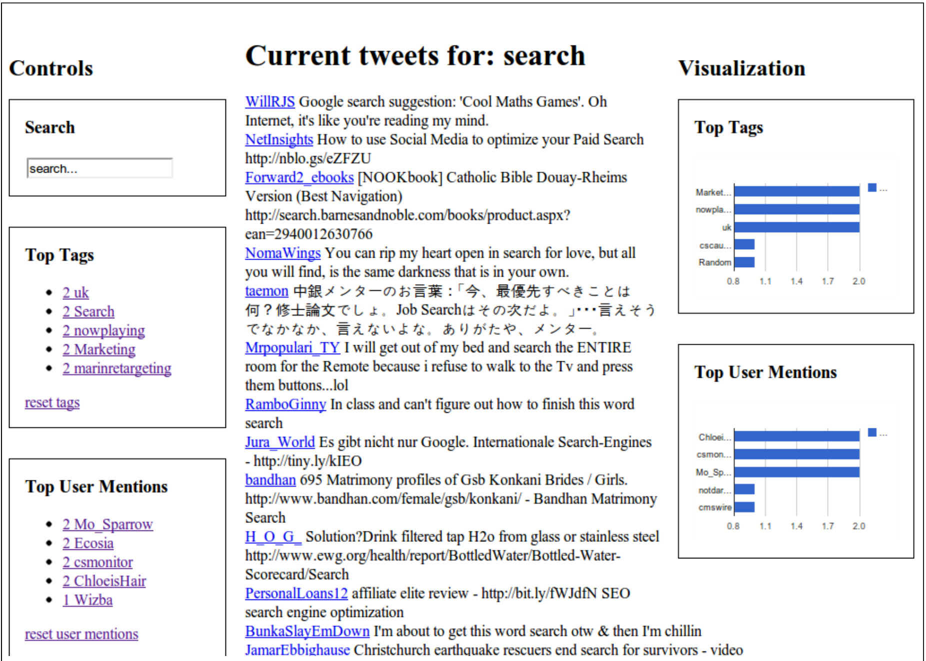
[2]http://en.wikipedia.org/wiki/Twitter#Messages

## How to use it

*Twitter Search Aggregation* uses node.js in order to start a server which communicates via Twitter. The following steps start the application:

1. Download and install node.js from http://nodejs.org/#download
2. Download the project from https://github.com/dasjo/twitter-search-aggregation
3. In terminal navigate to the project folder “twitter-search-aggregation”
4. Start the server: *node server.js <twitter_username> <twitter_password>*.
5. Open *index.html* with a WebSocket compatible browser (Chrome or Webkit nightly).

If everything worked, your browser will show the *Twitter Search Aggregation* application and perform a default search on the string “search”. The screen looks like the following:



**Controls**

**Search**

search...

**Top Tags**

- 2 uk
- 2 Search
- 2 nowplaying
- 2 Marketing
- 2 marinretargeting

reset tags

**Top User Mentions**

- 2 Mo_Sparrow
- 2 Ecosia
- 2 csmonitor
- 2 ChloeisHair
- 1 Wizba

reset user mentions

**Current tweets for: search**

WillRJS Google search suggestion: 'Cool Maths Games'. Oh Internet, it's like you're reading my mind.
NetInsights How to use Social Media to optimize your Paid Search http://nblo.gs/eZFZU
Forward2_ebooks [NOOKbook] Catholic Bible Douay-Rheims Version (Best Navigation) http://search.barnesandnoble.com/books/product.aspx?ean=2940012630766
NomaWings You can rip my heart open in search for love, but all you will find, is the same darkness that is in your own.
taemon 中銀メンターのお言葉：「今、最優先すべきことは何？修士論文でしょ。Job Searchはその次だよ。」･･･言えそうでなかなか、言えないよな。ありがたや、メンター。
Mrpopulari_TY I will get out of my bed and search the ENTIRE room for the Remote because i refuse to walk to the Tv and press them buttons...lol
RamboGinny In class and can't figure out how to finish this word search
Jura_World Es gibt nicht nur Google. Internationale Search-Engines - http://tiny.ly/kIEO
bandhan 695 Matrimony profiles of Gsb Konkani Brides / Girls. http://www.bandhan.com/female/gsb/konkani/ - Bandhan Matrimony Search
H_O_G_ Solution?Drink filtered tap H2o from glass or stainless steel http://www.ewg.org/health/report/BottledWater/Bottled-Water-Scorecard/Search
PersonalLoans12 affiliate elite review - http://bit.ly/fWJdfN SEO search engine optimization
BunkaSlayEmDown I'm about to get this word search otw & then I'm chillin
JamarEbbighause Christchurch earthquake rescuers end search for survivors - video

**Visualization**

**Top Tags**

**Top User Mentions**

*Illustration 1: Twitter Search Aggregation screenshot*

The user interface consists of the following parts:

### Current tweets for: <search_string>

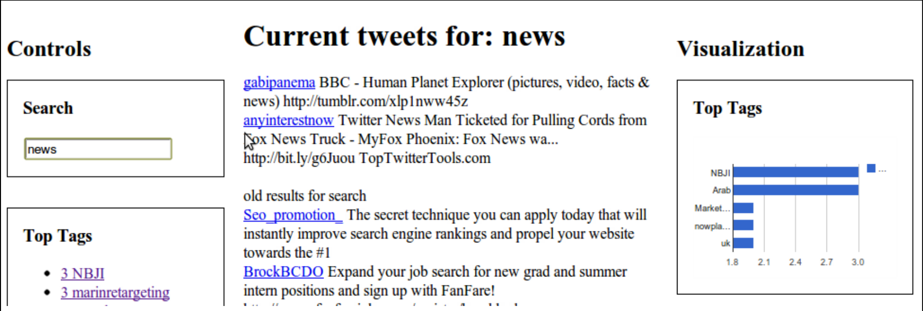
The current tweets section in the center of the application displays the live-stream of twitter messages matching the current search string. New items will appear at the top of the list, moving older items downwards. Items exceeding the limit of 15 maximum items will disappear at the lower end of the tweets list. As stated, by default *Twitter Search Aggregation* will look for tweets matching “search”. The search string can be modified using controls explained subsequently.

### Controls

The controls section appears at the left column of the application. It provides interactive controls to interact with the application.

### Search

A search box allows to change the current search term. Entering text into the provided field and submitting via the *Enter key* change the current search term. The current tweets list will also indicate so and insert a spacer in order to visually sparate tweets matching the old search term from the new results matching the current search term.



Controls

Search

news

Top Tags

- 3 NBJI
- 3 marinretargeting

Current tweets for: news

gabipanema BBC - Human Planet Explorer (pictures, video, facts & news) http://tumblr.com/xlp1nww45z
anyinterestnow Twitter News Man Ticketed for Pulling Cords from Fox News Truck - MyFox Phoenix: Fox News wa... http://bit.ly/g6Juou TopTwitterTools.com

old results for search
Seo_promotion_ The secret technique you can apply today that will instantly improve search engine rankings and propel your website towards the #1
BrockBCDO Expand your job search for new grad and summer intern positions and sign up with FanFare!

Visualization

Top Tags

*Illustration 2: The current search term was changed from "search" to "news"*

### Top Tags & Top User Mentions

These sub-sequent sections within the controls area under the search box are the heart of *Twitter Serach Aggregation*. They display *tags* and *user mentions* which appear the most within all tweets that the performed twitter search returned. Clicking on a *tag* or *user mention* will change the current search string to the name of the clicked item. In addition, the *reset tags* and *reset user mentions* links enable the user to start from zero via resetting aggregation values of the given section.

### Visualization

The visualization section on the right column displays top tags and top user mentions accordingly to their counter-part within the controls section using *Google Charts*.

# Technology

The following chapter discusses technology decisions, the actual realization and open issues & possible improvements for *Twitter Search Aggregation*.

## Decisions

Initially, the plan was to investige in Drupal 7-related technologies for querying different microblogging search backend including twitter and status.net due to the author's personal interest and experience with the content management framework/system Drupal. For the time of investigation, most Drupal modules connecting Drupal to twitter hadn't been available for Drupal 7 yet. Furthermore, from a personal perspective, Drupal is more architectured to be full-fledged content management system plus great framework capabilities which makes it simply “too heavy” for developing light-weight tools like the Twitter Search Aggreagtion prototype is thought to be.

As a light-weight alternative, Twitter Search Aggregation is realized as a Node.js WebSockets HTML5 application due to the following thoughts:

- Node.js has proven to be a scaleable approach to developing network programs on a low-level. As Wikipedia states: “Node.js is an event-driven I/O framework for the V8 JavaScript engine on Unix-like platforms. It is intended for writing scalable network programs such as web servers”[3].
- DevelopmentSeed, a sucessful washington-based web shop that built popular Drupal tools as Open Atrium and Managing News, recently switched to node.js for realizing their recent mapping solutions.[4] This doesn't mean, they generally state node.js being superior to Drupal, but recognizing that node.js is a more light-weight approach that allows them tackle their challenges more accordingly.
- With the existence of the “*Twitter Node.js WebSockets Example*” a basis for developing *Twitter Search Aggregation* using the selected technologies has been found.

3 http://en.wikipedia.org/wiki/Nodejs
4 http://drupalradar.com/development-seed-sell-drupal-work-phase-2

**Realization**

The following diagram illustrates the architecture of *Twitter Search Aggregation*:

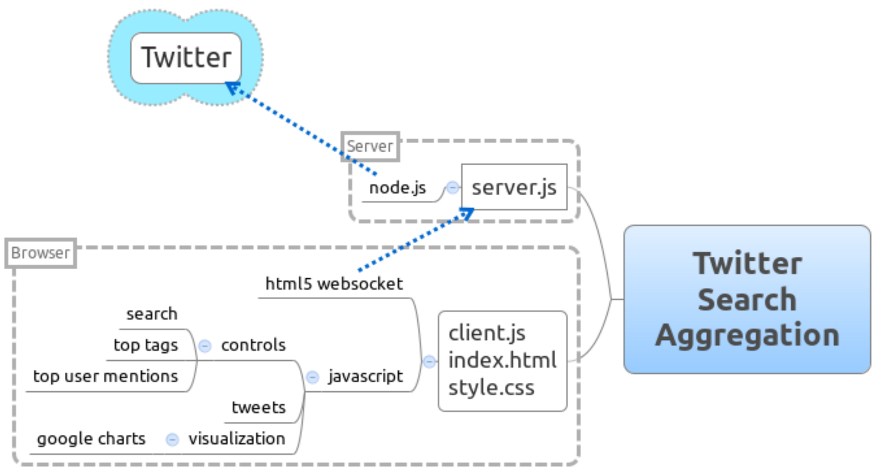


*Illustration 3: Twitter Search Aggregation Architecture*

Section *How to use it* describes the steps needed in order to start and use the application. Let's discuss them from a technological point of view by stating some of the core code snippets:

*Start the server: node server.js <twitter_username> <twitter_password*

```
// Websocket TCP server
ws.createServer(function (websocket) {
  clients.push(websocket);

  websocket.addListener("data", function(message){
    ...
  });

  websocket.addListener("connect", function (resource) {
    // emitted after handshake
    sys.debug("connect: " + resource);

  }).addListener("close", function () {
    // emitted when server or client closes connection
    clients.remove(websocket);
    sys.debug("close");
  });
}).listen(8081);
```

This will execute the server part of Twitter Search Aggregation. It creates a Websocket TCP server avaiting connecting and data from clients.

*Open index.html with a WebSocket compatible browser (Chrome or Webkit nightly).*

The *index.html* Browser page is opened, displays all static html content and loads the *client.js* JavaScript file which contains all logic to perform the real-time twitter aggregation performed by *Twitter Search Aggregation*.

```
ws = new WebSocket("ws://localhost:8081/");
ws.onmessage = processServerInput;
ws.onclose = function() {
  ...
};
ws.onopen = function() {
  invokeSearch("search");
};
```

This connects the HTML5 client to the node.js server using WebSockets and issues the default search for the string “search”.

The server.js reacts on the client's search request via the websocket listener:

```
websocket.addListener("data", function(message){
  if(tweetsResponse) {
    tweetsResponse.removeListener("data", responseListener);
  }

  twitter = http.createClient(80, "stream.twitter.com");
  request = twitter.request("GET", "/1/statuses/filter.json?track=" + message, headers);
  request.addListener('response', responseFunction);
  request.end();
  console.log("Now searching for: " + message);

});
```

This replaces stops any current search from being executed and creates a new twitter search request using *Twitter Search API*. Twitter's response is a stream which will be handled by the responseFunction and the responseListener respectively:

```
responseListener = function(chunk) {
  // Send response to all connected clients
  tweets += chunk;

  if((tweets.split(/{/g).length - 1) == (tweets.split(/}/g).length - 1)) {
    console.log(tweets.substring(0, 50));

    clients.each(function(c) {
      c.write(tweets);
    });

    tweets = "";
  }
}
```

The responseListener concatenates the response and when detecting a complete tweet, it will send the tweet to all clients listening.

Again, on the client side a listener function reacts on tweets sent by the server:

```
function processServerInput(evt) {
  data = eval("(" + evt.data + ")");
  //data = jQuery.parseJSON("(" + evt.data + ")");

  var hashtags = data.entities.hashtags;
  processCounter(hashtags, "tags");

  var user_mentions = data.entities.user_mentions;
  processCounter(user_mentions, "user_mentions");

  //add tweet to list
  addTweet("<div class='content'><a class='main-screenname' href='http://www.twitter.com/" +
data.user.screen_name + "/status/" + data.id + "' target='_blank'>" + data.user.screen_name +
"</a> " + data.text + "</div>");
}
```

client.js contains more JavaScript code in order to display tweets to the user, as the aggregation tasks for counting tags and user mentions. Also it includes a charting functionality using Google Charts to draw diagrams which display the top tags and top user mentions to the user.

## Open Issues & Possible Improvements

As stated in the introduction, the current implementation of Twitter Search Aggregation isn't intended to be perfect at all. While it basically works, there exist known design issues and bugs which are going to be discussed in this section.

### Tweet parsing

The mechanism to parse tweets isn't perfect at all. The problem is, that the chunks received from twitter don't always represent whole tweets, but they may be truncated.

The current approach is to ensure completeness of chunks on the server-side before forwarding them to the client. This guarantees the client that it can transform the tweets-stream to JSON always. The downside is that some tweets will be “cached” by the server before forwarding them to the client due to the following procedure applied. The server will accumulate the tweet chunks it receives until it detects the same number of brackets being opened as closed.

If for example twitter sends the server one complete tweet plus the beginning of another tweet within a single chunk of HTML response, the current implementation will not forward the first complete tweet to the client until it receives the missing part of the second one, fulfilling the condition “number of opening brackets equals number of closing brackets”.

A more intelligent approach would forward complete tweets at receival.

### Core Aggregation vs Visualization Aggregation

Currently, two separate algorithms calculate the number of hashtags and user mentions. The first is the “core aggregation” component which gears the top tags and top user mentions control units at the left column. The visualization add-on is based on Google Charts and therefore leverages a Charts-specific storage for aggregating tags and user mentions. The results of both approaches seem to differ in some cases which means that at least one of both algorithms is incorrect.

A superior implementation of aggregating would leverage a central datastore for storing the aggregation data which then drives the different displaying components as the controls panel and google charts.

### Asynchronous Code

Both server.js and client.js should be refactored to support asynchronous calls for one server supporting multiple clients at the same time. This hasn't been respected thoughoutly for the current, prototypic implementation.

### Connect additional external services

It would be interesting to connect further web services to display contextual information specially for tags. One could for example think of displaying Wikipedia entries related to the most popular twitter hashtag for the search, currently performed.

### Twitter limitations

If performing too many searches within a short time-frame, twitter blocks the user account temporarily. Excessive use might even lead to a permanent block.

Google Charts implications

Updating the diagrams using Google Charts API causes the browser to scroll down, degrading user experience.

## Conclusions

Twitter Search Aggregation is a functional prototype which sucessfully illustrates how a live tracking of twitter searches with aggregation of entities as hashtags and user mention works. It leverages exiting new technologies as node.js and HTML5 websockets. While implementing the application required investigation of new technologies it was a good experience especially to enjoy the fast workflow, developing under node.js provides. Re-starting the server works within seconds and on the client-side also reloading the client browser page is really quick.