Measuring Public Knowledge of Trending Topics

Archishman Mitra

DSCS 6020 Spring 2016
Term Project

April 2016

Overview :

What do we do when a friend asks us if we have heard about this topic which went 'viral'? We feel like a fool that we don't know about it and immediately Google it and click on the first link which most often than not is a Wikipedia page. This is just a theory. The objective of this project is to mine, clean and store data which can be used to support, validate and further this theory.

Phase 1: Trending Topic

The Twitter RESTful API is used to find the top 50 current trending topics. The #Hashtag or the ‘Trending Topic’ gathered is then used to create a searchable item to be used in the next phase.

Phase 2: Wikipedia Article Search

The WIkipedia Opensearch API is an open source project which can be used to find articles in Wikipedia. This is used to search the ‘searchable item’ created in the last phae to find the relevant articles.

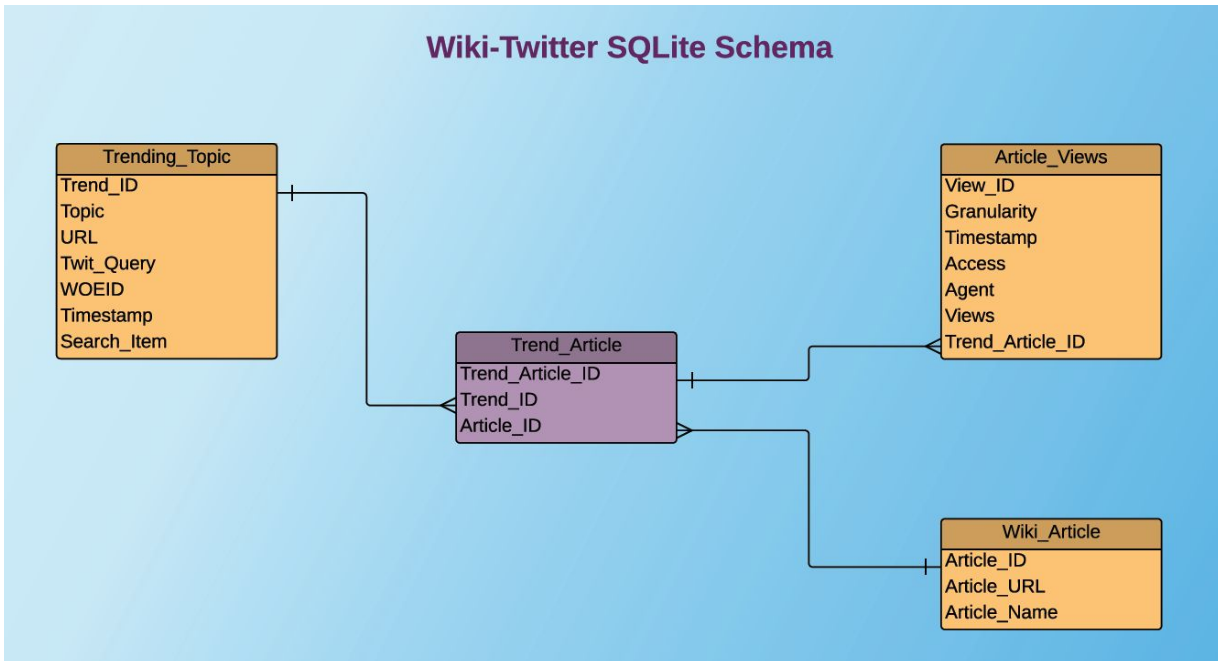
Phase 3: Article View Count

The MediaWiki API provides a very useful ‘pageview’ programmable URL. The page view count for articles found in Phase 2 is mined using this.

Database :

At first it seemed like a NoSQL databases capable of handling JSON objects would be the appropriate database for this. However, the requirement of structuring the data for analysis, the availability of libraries to structure data and a necessity to have a relational model led to choosing a relational DB. Also, notwithstanding the retrieval functions and schema definition requirements, SQLite was chosen.

The schema has four tables : Trending_Topic, Trend_Article, Article_Views and Wiki_Article. The Trend_Article is a junction table which stores the many to many relationship between Trending_Topic and Wiki_Article. It also provides a Foreign Key to Article_Views. A more detailed explanation on the contents of each table is discussed in the below as and when the requirement arises. Below is the schema implemented for this project.



Phase 1:

1. **Twitter RESTful API:** Once a successful connection is established with the Twitter API and the app has been authorized, getTrends() function available through the TwitteR package is used to pull the Top 50 current Trending Topics. This get request allows a WOEID to be passed to restrict the return to a particular location. In this case, WOEID = 23424977 is used which is for USA. A timestamp is added to indicate the time this

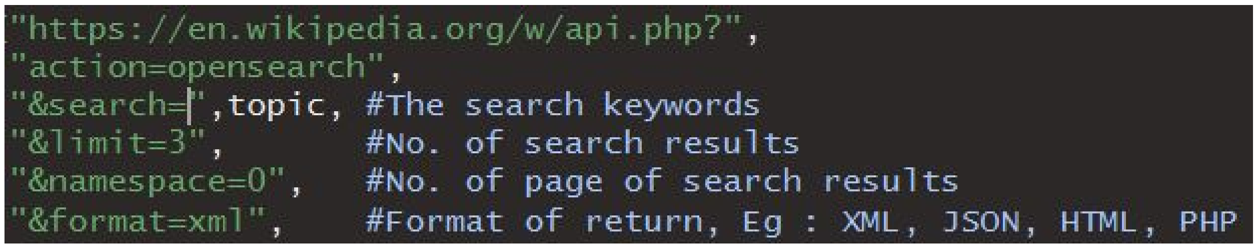
request was performed. Twitter caches this lists for the next 5 minutes which means if another request is made within 5 minutes the same content would be pulled.

2. **Creating a Searchable item:** The structure of the string holding the Trending Topics is varied making the conversion to a Searchable item really interesting. Some of the examples: #GameofThrones, Byron Scott, #IfYouDontKnowByNowThat, #EBI6 Catering to all cases would be a herculean task and would constitute a project in itself. Hence, a simple yet effective parsing mechanism was devised which readily converts almost 80% of the trending topic string to Searchable items.
3. **Database Implementation:** The dataframe build from the above two stages is then added a unique ID. This ID becomes the Primary key of table Trending_Topics. The table with the following columns : Trend_ID, Topic, URL, Twit_Query, WOEID, timestamp, Search_Item is dumped in the WikiTwitter database.

Phase 2 :

This is created as a function to enable the future consumer to first look through the Trending Topics from Phase 1 and decide which topic is of interest. It takes inputs of trend_id and topic.

1. **Wikipedia Search API:** The Wikipedia Search API is a programmable URL which has the following structure and programmable query strings:



```
"https://en.wikipedia.org/w/api.php?",
"action=opensearch",
"&search=",topic, #The search keywords
"&limit=3",        #No. of search results
"&namespace=0",    #No. of page of search results
"&format=xml",     #Format of return, Eg : XML, JSON, HTML, PHP
```

Here, the ‘topic’ is the Searchable item from Phase 1. The initial intention was to use format as JSON but the returned blob of data was very difficult to parse. On the contrary, the XML format returned an object which was much easier to parse. A parsing mechanism was built to extract the article names from the returned object while also preserving the URL for each article embedded in the returned object.

2. **Data Integrity:** As this phase searches for article names and saves them, it was imperative that the whole process for water-tight in terms of maintaining data integrity. A check was introduced to check if a particular article name is present in the database before putting that again in the database. Equally important was to create a placeholder for cases when no articles were found.

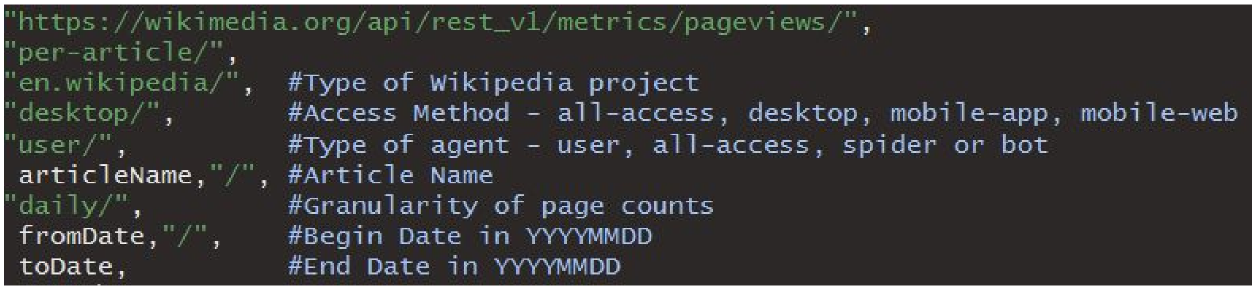
3. **Database Implementation:** The dataframe build in the first stage of this phase is added an unique ID and dumped in the database as table Wiki_Article with column names Article_ID, Article_URL & Article_Name. The articles found in this phase has a many to many relationship with the trends from Phase 1. Therefore, a junction table Trend_Article encompassing this concept is created and populated at this phase. It has the following column names: Trend_Article_ID, Trend_ID, Article_ID.

Phase 3 :

This is created as a function to enable the consumer to look through the articles found in the previous phase and understand view counts of which article, of which duration and of which granularity will be relevant for his/her analysis.

1. **MediaWiki Pageview API:** The MediaWiki API has the following structure and query strings.This returns a JSON object with the page view counts for the query string

passed duration. This JSON object is then converted into a dataframe to be put in a



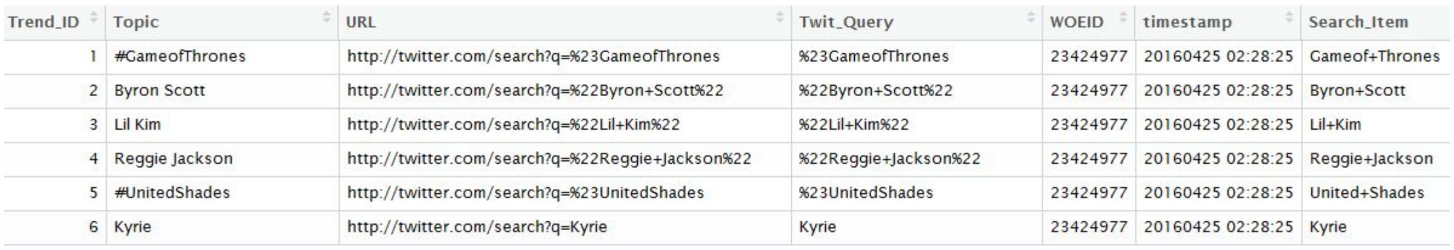
```
"https://wikimedia.org/api/rest_v1/metrics/pageviews/",
"per-article/",
"en.wikipedia/",   #Type of Wikipedia project
"desktop/",        #Access Method - all-access, desktop, mobile-app, mobile-web
"user/",           #Type of agent - user, all-access, spider or bot
 articleName,"/",  #Article Name
"daily/",          #Granularity of page counts
 fromDate,"/",     #Begin Date in YYYYMMDD
 toDate,           #End Date in YYYYMMDD
```

table.

2. **Database Implementation:** A column of Trend_Article_ID is added to the dataframe from the previous stage to give reference to the trending topic and article the record pertains to. This is then put in a table Article_Views with columns View_ID, Granularity, Access, Agent & Trend_Article_ID.

Data Population and Retrieval :

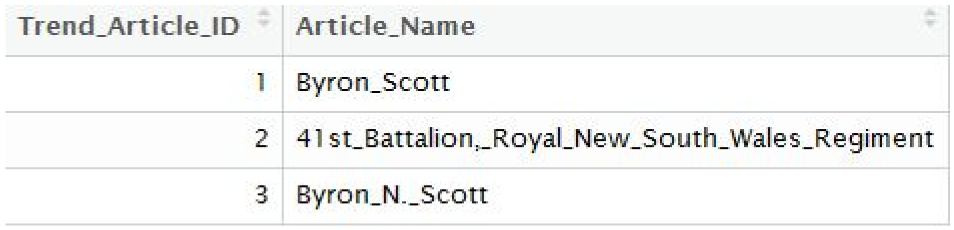
1. **Trending Topic:** A SQL query to retrieve for a particular duration the previously saved trending topic data from Phase 1. A Sample output:



Trend_ID	Topic	URL	Twit_Query	WOEID	timestamp	Search_Item
1	#GameofThrones	http://twitter.com/search?q=%23GameofThrones	%23GameofThrones	23424977	20160425 02:28:25	Gameof+Thrones
2	Byron Scott	http://twitter.com/search?q=%22Byron+Scott%22	%22Byron+Scott%22	23424977	20160425 02:28:25	Byron+Scott
3	Lil Kim	http://twitter.com/search?q=%22Lil+Kim%22	%22Lil+Kim%22	23424977	20160425 02:28:25	Lil+Kim
4	Reggie Jackson	http://twitter.com/search?q=%22Reggie+Jackson%22	%22Reggie+Jackson%22	23424977	20160425 02:28:25	Reggie+Jackson
5	#UnitedShades	http://twitter.com/search?q=%23UnitedShades	%23UnitedShades	23424977	20160425 02:28:25	United+Shades
6	Kyrie	http://twitter.com/search?q=Kyrie	Kyrie	23424977	20160425 02:28:25	Kyrie

2. **Article Search:** Once a trending topic of substantive interest is identified the function ArticleSearch defined in Phase 2 is used to find articles concerning the topic. Next, a

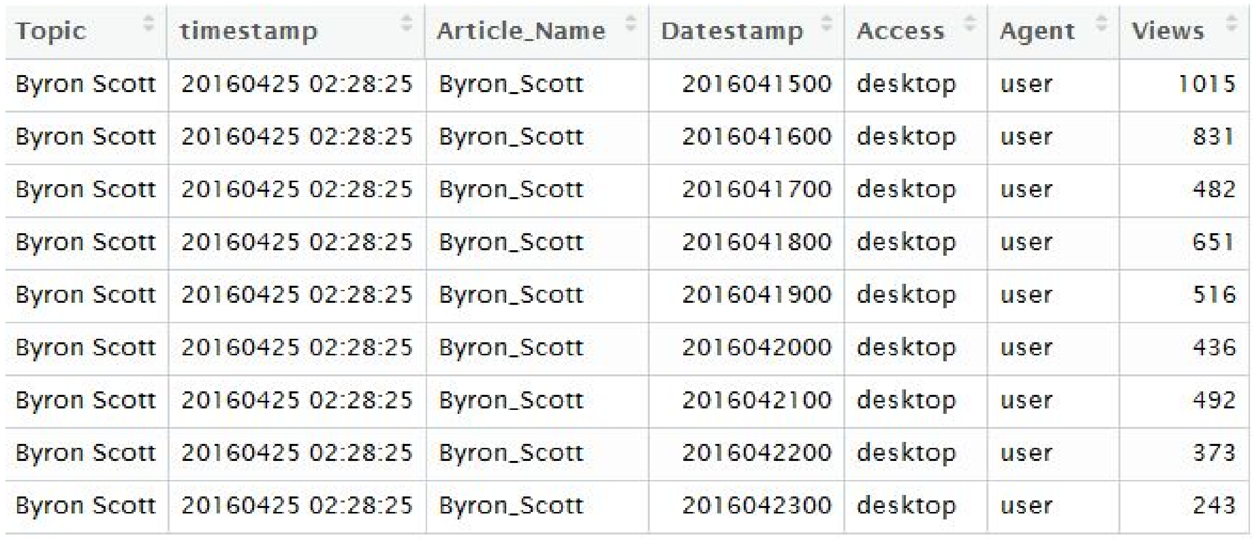
SQL query is used to retrieve those article names. A sample output :



Trend_Article_ID	Article_Name
1	Byron_Scott
2	41st_Battalion,_Royal_New_South_Wales_Regiment
3	Byron_N._Scott

3. **Page Count:** The consumer next decides the article for which view counts are to be gathered. The duration for which the view counts are required along with the article name and its trend_article_id is passed to the function PageCountfun defined in Phase 3.

4. **WikiTwitter:** A SQL using JOINs is used to build a dataframe which has the view counts, article name, trend topic and other details to be used for analysis. A sample output:



Topic	timestamp	Article_Name	Datestamp	Access	Agent	Views
Byron Scott	20160425 02:28:25	Byron_Scott	2016041500	desktop	user	1015
Byron Scott	20160425 02:28:25	Byron_Scott	2016041600	desktop	user	831
Byron Scott	20160425 02:28:25	Byron_Scott	2016041700	desktop	user	482
Byron Scott	20160425 02:28:25	Byron_Scott	2016041800	desktop	user	651
Byron Scott	20160425 02:28:25	Byron_Scott	2016041900	desktop	user	516
Byron Scott	20160425 02:28:25	Byron_Scott	2016042000	desktop	user	436
Byron Scott	20160425 02:28:25	Byron_Scott	2016042100	desktop	user	492
Byron Scott	20160425 02:28:25	Byron_Scott	2016042200	desktop	user	373
Byron Scott	20160425 02:28:25	Byron_Scott	2016042300	desktop	user	243

Insights :

1. **Learnings:** While the project is a lot more streamlined now, it ran into some very interesting problems. Building the parsing mechanisms involved exploring Regular Expressions and XML structures. Choosing and implementing a database was a huge pain point. As mentioned above, a NoSQL database, more specifically MongoDB was seriously considered and almost implemented before making a U-turn in favour of SQLite because of the data integrity and relational modelling issues. The original proposal of using was dealt a major blow when it completely stopped taking requests other than web-agents. Thankfully, some exploring led to the very convenient MediaWiki viewcount API.
2. **Future Work:** The trending topics used for this project is limited to the just the currently available ones. However, if the Phase 1 is run periodically, a rich repository consisting of trending topics from every 5 minutes can be built. This repository can then be used to find trending topic at weekly, monthly etc granularity levels. In the same way, this added usefulness can also be extended to Phase 3, where the granularity of the view count data can be controlled to match to new functionality in Phase 1. Besides this, the potential of the programmable URLs is worth exploring. From the database perspective, more data integrity checks can be introduced in both Phase 1 and Phase 3.

References :

1. https://dev.twitter.com/overview/documentation
2. https://www.mediawiki.org/wiki/API:Main_page
3. https://www.mediawiki.org/wiki/RESTBase
4. https://wikimedia.org/api/rest_v1/?doc
5. http://tools.wmflabs.org/pageviews/#project=en.wikipedia.org&platform=all-access&agent=user&range=latest-20&pages=Cat|Dog