

# CSE 435/535 Information Retrieval Fall-2020 University at Buffalo (SUNY)

Instructor: - Dr. Rohini K. Srihari

Project One : Content Curation, Ingestion, Tokenization and  
Indexing

**Due Date : 19th September 2020, 23:59 EST/EDT**

1. Introduction
  2. Major Tasks and Challenges
  3. Project requirements
  4. Submitting your project
  5. Grading
  6. Pre-requisites
  7. Collecting and Storing Twitter Data
    - 4.1 Authentication
    - 4.2 Streaming and REST APIs
    - 4.3 User Timeline
    - 4.4 Twitter Clients
  8. Indexing
    - 5.1 Solr terminology
    - 5.2 Indexing strategies
  9. FAQs
- Appendix
- Character encoding
  - Emoticons, Emojis and Kaomoji

## 1. Introduction

The primary purpose of this project is to introduce students to the different technical aspects involved in this course and subsequent projects. By the end of this project, a student would have achieved the following:

- Setting up an AWS account and a simple EC2 instance in AWS
- Learning about the Twitter API, and querying twitter using keywords, language filters and retrieving tweets from user timeline. Make sure to store your data as you will be using this data in the final project.
- Setting up a Solr (a fully functional, text search engine in Java) instance - understand basic Solr terminology and concepts. Refer to [Solr Reference Guide 8.1](#) which contains detailed explanation of operations you will be using in all the projects in this course.
- Understanding fundamental concepts of IR such as tokenization, indexing, search etc.
- Indexing thousands of tweets in multiple languages

The specific challenges in completing this project are as given below:

- Curate the Twitter personalities who hold significant influence on their followers and write script to retrieve replies from their tweets.
- Curate the keywords and hashtags for effective data collection which can be used for downstream tasks of classification, sentiment analysis etc.
- Working with multi-lingual data, and learning to index and tokenize them for search.
- Correctly setup the Solr instance to accommodate language and Twitter specific requirements.

The rest of this document will guide you through the necessary setup, introduce key technical elements and then finally describe the requirements in completing the project. This is an individual project and all deliverables **MUST** be submitted by **19th September 23:59 EST/EDT**.

**Note:** The data collected and work done in this project will act as foundation for Project 4 to analyze reaction to government policies on COVID pandemic on Twitter.

## 2. Major Tasks and Challenges

Major Task	Challenges
Twitter Developer Account	Takes maximum 2-3 days to setup.

AWS registration and EC2 setup	
Solr Setup	
Solr	<ul style="list-style-type: none"> <li>• Getting familiar with Schema files, filters and analyzers.</li> <li>• Understanding Solr Admin UI and how indexing works.</li> <li>• Understanding how queries work on Solr Admin UI.</li> </ul>
Script for crawling and processing raw tweets	<ul style="list-style-type: none"> <li>• Retrieving tweets from user timeline.</li> <li>• Search using hashtags and keywords.</li> <li>• Search while being within Twitter rate limit. Read the rate limit guidelines carefully.</li> <li>• Getting familiar with JSON content and extracting fields needed for this project.</li> <li>• Finding ways to handle emoticons, dates and multilingual data.</li> </ul>
Crawling 40,000 tweets with various requirements	<ul style="list-style-type: none"> <li>• Starting early to meet all minimum requirements.</li> <li>• Handling duplicate tweets</li> <li>• Handling RTs</li> </ul>

### 3. Project requirements

We now describe the actual project. As mentioned before, the main purpose of this project is to index a reasonable volume of tweets and perform rudimentary data analysis on the collected data. This year, projects will be focused on COVID-19 pandemic which quickly evolved and made a widespread impact. Goal of this project is to perform social media mining to retrieve and index tweets from government officials across different geographies. Data collected and indexed in this project will be used for downstream tasks and analyses in Project 4.

In this project, we are specifically interested in tweets from following countries:

- USA
- India
- Italy

Your dataset should be multilingual and contain following languages

- English
- Hindi
- Italian

We want you to collect tweets that meet the following criteria:

- (i) Tweets reflecting citizen reaction to government policy related to Covid
- (ii) Additionally, collect tweets from at least 3 persons of interest (POI) per country who are involved in policy making. These POIs can be the government health agency, president/prime minister of the country, or some other government leader.

You can refer to this website to find popular POIs on Twitter <https://www.socialbakers.com/statistics/twitter/profiles>. Additionally, you can use CIA factbook for each country to know more about their ruling government. (Eg: CIA factbook for Italy can be found here: <https://www.cia.gov/library/publications/the-world-factbook/geos/it.html>)

We now describe the two main tasks with additional content curation requirements.

### **Task 1: Content Curation and Ingestion**

Curate the required set of query terms, language filters, person of interest and combinations thereof to crawl and index tweets, subject to the following requirements:

1. At least 40,000 tweets in total with not more than 15% being retweets.
2. At least 5,000 tweets per language i.e, English, Hindi and Italian
3. At least 5,000 tweets per country
4. At least 500 tweets per person of interest (timeline). [Note that Twitter allows max 3200 recent tweets to be extracted from a person's account.](#)
5. During at least 5-day window in each country:
  - a. At least 1 tweet by each of the POIs per day
  - b. At least 3,000 tweets collected per day which are about reactions of general public to government's policies on COVID. **Tip:** Curate query terms(hashtags/keywords/mentions) which can extract tweets relevant to both COVID **and** government policy (eg: 'covid AND government').

- c. This means that for the collected data, the tweet date must have at least 5 distinct values, and for each such day there must be at least 3,000 tweets.
- d. Essentially, you cannot collect say 20,000 tweets on one day and split the rest between other two days.
- e. Keep in mind that the search index has a 7-day limit. In other words, no tweets will be found for a date older than one week.
- f. Additionally, you cannot collect tweets talking just about COVID in general times (for eg: “I hope #COVID19 goes away before The Batman is released next year!!!” is not a reaction to government’s policy).

**Tip:** Best way to come up with a good list of hashtags/keywords is to browse the Twitter timeline of the POI you have selected.

**Tip2:** Use Twitter’s advanced search to formulate queries: <https://twitter.com/search-advanced?lang=en>

**Tip3:** Go through these rules on GET request: <https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets>.

Note that the above are the minimum requirements. You are free to crawl tweets in other languages outside this list (or crawl tweets without a language filter for example) as long as the above requirements are met. Further, this data would be validated against your Solr index. Thus, based on how you setup your indexing, you may lose some tweets and/or have duplicates that may reduce your indexed volumes. Hence, it is encouraged that you accommodate some buffer during the crawling stage.

Once you have collected your tweets, you would be required to index them in a Solr instance. You would need to tweak your indexing to adhere to two distinct sets of requirements - language specific and Twitter specific as described below.

## **Task 2: Tokenization and Indexing**

Index the collected tweets subject to the following requirements:

1. Person of Interest: Name and Ids of at least 9 persons of interest
2. Country: one amongst USA, India and Italy

4. One copy of the tweet text that retains all content (see Section 4) irrespective of the language. This field should be set as the default field while searching.
5. Language of the tweet (as identified by Twitter) and a language specific copy of the tweet text that removes all stopwords (language specific), punctuation, emoticons, emojis, kaomojis, hashtags, mentions, URLs and other Twitter discourse tokens. Thus, you would have at least four separate fields that index the tweet text, the general field above plus three for each language. For any given tweet, only two of the four fields would have a value.
6. Separate fields that index: hashtags, mentions, URLs, tweet creation date, emoticons+ (emoticons + emojis+ kaomojis)

Some naming conventions are required to be used as described below:

- `poi_name` : Screen name of one of the 9 persons of interest. Null if the tweet is from general public.
- `poi_id` : User Id of one of the 9 persons of interest. Null if the tweet is from general public.
- `country` : One of the 3 countries
- `tweet_text` : Default field
- `tweet_lang` : Language of the tweet from Twitter as a two letter code.
- `text_xx` : For language specific fields where xx is at least one amongst en (English), hi (Hindi) and it (Italian)
- `hashtags` : if there are any hashtags within the tweet text
- `mentions` : if there are any mentions within the tweet text
- `tweet_urls` : if there are any urls within the tweet text
- `tweet_emoticons` : if there are any emoticons within the tweet text
- `tweet_date` : Tweet creation date rounded to nearest hour and in GMT

#### 4. Submitting your project

We would use the CSE submit script. You would be required to submit two text files:

- *project1.txt*: It should only contain the IP address of your EC2 instance followed by your UBIT name, separated by a **tab**
- *query.txt*: It should contain all the hashtags/keywords you used to create your queries for data collection.

For submission, follow these steps:

1. Run your Solr instance on port 8984. Make sure that the port is accessible during grading duration.
2. Name your Solr core as IRF20P1. So if the IP address of your EC2 instance is aa.bb.cc.dd, the Solr query page should be accessible as <http://aa.bb.cc.dd:8984/solr/#/IRF20P1/query>

**Note that grading is automated and all naming conventions must be followed strictly.**

## 5. Grading

This project is worth a total of 10 points, these are distributed as follows:

Task	Criterion	Description	Points
Task 1	Tweet Volumes	Validate at least 40,000 tweets	1
	Language Volumes	Validate at least 5,000 tweets for en, hi, it	0.5
	Country Volumes	Validate at least 5,000 tweets per country	0.5
	Date criteria	Validate at least 5 days with at least 3,000 tweets per country	1
	Person of Interest criteria	Validate: <ul style="list-style-type: none"><li>• at least 9 persons of interest chosen</li><li>• at least 500 tweets retrieved from person of interest's timeline</li></ul>	1
	Retweet Counts	Validate at most 15% retweets	0.5
Task 2	Sanity	Validate Solr instance runs + can run some queries	2
	Schema Validation	All fields are named as required, contain values as required, etc.	1.5



	Data Sanity	Analysis based on query terms chosen to extract reactions to government policies on COVID.	2
--	-------------	--	---

We will run the grading script once for a dry run on the date which will be specified later. This will allow you to perform a sanity check of your submission. The grades of dry run will **not** be counted towards final grades.

The final grading script will be triggered right at midnight and thus any late submissions will not be considered.

## 6. Pre-requisites

The first thing you need is to setup AWS EC2 instance where you will be hosting your project; Twitter developer account to retrieve tweets; and Solr instance for indexing operations. Please refer “SetupGuidebook.pdf” to guide you through all the setups.

**Note:** Windows users will also need to install open source client Putty and file transferring tool FileZilla to access EC2 instance.

## 7. Collecting and Storing Twitter Data

We will be using UBbox to store data. Every student has access to their UBbox account which has unlimited space to store your data. UBbox also provides version control and ability to share your data with other students. Sharing and collaboration will come in handy during group projects. **Note that for project 1, sharing tweets will not be allowed.** We will include various checks in grading script to make sure all the students have collected their own crawled tweets. Another reminder that Academic Integrity is an important issue and department will not be lenient, if the policy is breached.

Please access your UB box account here  
<https://www.buffalo.edu/ubit/ubbox.html>

**Note:** Make sure not to post AWS key or any other sensitive information on your UBbox.

For tweets collection, there are three main elements that you need to know with regards to using the Twitter API : Authentication, Streaming vs REST APIs and Twitter Clients.

#### **4.1 Authentication**

Twitter uses OAuth to authenticate users and their requests to the available HTTP services. The full OAuth documentation is long and exhaustive, so we only present the relevant details. Please refer “SetupGuidebook.pdf” to setup Twitter developer account and create access tokens.

#### **4.2 Streaming and REST APIs**

We are only concerned about querying for tweets i.e. we do not intend to post tweets or perform any other actions. To this end, Twitter provides two types of APIs : REST (which mimics search) and Streaming (that serves “Live” data).

You are encouraged to experiment with both to see which one suits your needs better. You may also need a case by case strategy - search would give you access to older data and may be more useful in case sufficient volumes don’t exist at a given time instant. On the other hand, the Streaming API would quickly give you thousands of tweets within a few minutes if such volumes exist. Both APIs return a JSON response and thus, it’s important that you get yourself familiarized with the different fields in the response.

Please read up on the query syntax and other details here: <https://dev.twitter.com/>. You may be interested in reading up on how tweets can be filtered based on language and/or geolocation. These may help you in satisfying your language requirements fairly quickly.

Similarly, you can find documentation for the Streaming API at the same location. Since we are not worried about exact dates (but only recent dates), either of the APIs or a combination may be used. We leave it to your discretion as to how you utilize the APIs.

**Note:** Twitter specifies rate limits on standard API for GET (read) endpoints. Please refer <https://developer.twitter.com/en/docs/basics/rate-limits>.

### 4.3 *User Timeline*

In this project, you are required to retrieve most recent tweets from the timeline of influential personalities and you can do that by using Twitter's REST APIs. However, there is strict rate limiting on the number of tweets you can retrieve from the timeline. If these limits are not followed, you may end up getting your account suspended, therefore, we encourage you to please refer their API reference here : [https://developer.twitter.com/en/docs/tweets/timelines/api-reference/get-statuses-user\\_timeline.html](https://developer.twitter.com/en/docs/tweets/timelines/api-reference/get-statuses-user_timeline.html) to get information on the rate limits before you start your implementation.

### 4.4 *Twitter Clients*

Finally, there is a plethora of Twitter libraries available that you can use. A substantial (though potentially incomplete) list is present here: <https://developer.twitter.com/en/docs/developer-utilities/twitter-libraries>. You are welcome to use any library based on your comfort level with the library and/or the language used.

## 8. Indexing

Before we describe the indexing process, we introduce some terminology.

### 5.1 *Solr terminology*

- Solr indexes every **document** subject to an underlying **schema**.
- A schema, much akin to a database schema, defines how a document must be interpreted.
- Every document is just a collection of **fields**.
- Each field has an assigned primitive (data) **type** - int, long, String, etc.
- Every field undergoes one of three possible operations: *analysis*, *index* or *query*
- The analysis defines how the field is broken down into tokens, which tokens are retained and which ones are dropped, how tokens are transformed, etc.
- Both indexing and querying at a low level are determined by how the field is analyzed.

Thus, the crucial element is configuring the schema to correctly index the collected tweets as per the project requirements. Every field is mapped to a type

and each type is bound to a specific analyzer which contains tokenizers and filters. The schema.xml is responsible for defining the full schema including all fields, their types and analyzing, indexing directives.

Although a full description of each analyzer, tokenizer and filter is out of the scope of this document, a great starting point is at the following page : <https://cwiki.apache.org/confluence/display/SOLR/AnalyzersTokenizersTokenFilters> where you can find tips and tricks for all important elements you might want to use for this project. You are encouraged to start either in a schemaless mode or start with the default schema, experiment with different filters and work your way from there.

## 5.2 Indexing strategies

This is the part where students need to figure out the appropriate way to index their collected tweets. Overall, there are two overarching strategies that you must consider:

- Using out-of-the-box components and configure them correctly. For example, the *StopFilter* can be used to filter out stopwords as specified by a file listed in the schema. Thus, at the very minimum, you would be required to find language specific stopwords lists and configure the filters for corresponding type fields to omit these stopwords.
- Pre-processing tweets before indexing to extract the needed fields. For example, you could preprocess the tweets to introduce new fields in the json response as per project requirement. Here again, it is left to your choice of programming language and/or libraries to perform this task.

Solr supports a variety of data formats for importing data (xml, json, csv, etc). You would thus need to transform your queried tweets into one of the supported formats and POST this data to Solr to index.

## 9. FAQs

**Q:** If a tweet contains RT @ and the field value 'retweeted' is false, then is it considered a fresh tweet or retweet?

**A:** We would rely on the fields we have asked you to index to test for this. Since, retweeted isn't one of them, we would only look at the raw text.

**Q:** Why am I getting only 200 tweets when retrieving from user timeline?

**A:** Twitter allows 200 tweets for each request made for tweets on user timeline. You can make successive requests to retrieve more tweets, however, make sure you don't end up with redundant tweets in your final dataset.

**Q:** What is the best way to take care of redundant tweets?

**A: Hint:** Use “since\_id” and “max\_id”:

[https://developer.twitter.com/en/docs/tweets/timelines/api-reference/get-statuses-user\\_timeline.html](https://developer.twitter.com/en/docs/tweets/timelines/api-reference/get-statuses-user_timeline.html)

**Q:** I am getting errors while posting my JSON file on SOLR.

**A:** The first thing you should do in this case is to validate your JSON file. Please use [this](#) link to check whether your JSON file is valid or not. If your file is valid and you are still facing errors, use Piazza or office hours to ask TAs. Please provide full stack trace from the log files on Solr.

**Q:** What tools should I use to see my JSON file?

**A:** You can either use Sublime text editor or Notepad++

**Q:** Is the max 15% retweets restriction also applicable to the language specific tweets?

**A:** Applies to each of the four languages individually and at the global level.

**Q:** How should I process emoticons/emojis?

**A:** Use regular expressions, which are available quite easily, to filter out emojis/emoticons from you tweet text. You can also use user-defined dictionary for filtering out emoticons. There are lots of files available online containing emoticons.

**Q:** How to store date field in solr?

**A:** You can pre-process the date field from raw tweet into the format used by Solr.

It's a simple two-step process:

Step 1 : Convert the date returned by twitter from a string to a date object

Step 2 : Format the date into the format used by Solr.

## **Appendix**

### ***Character encoding***

One of the first issues you might encounter or have to deal with is character encoding. Every written character is encoded using one or the other character sets. Most programs (including your browser, favorite text editor, etc.) use some default encoding that is usually determined by your operating system. While using English or most Latin derived languages, the ASCII character set is sufficient and does not pose major problems. However, most other languages we have chosen for this project use extended character sets. For most scenarios, UTF-8 encoding might suffice.

However, there is the issue of UTF-16 encoding and Unicode characters. Some languages like Chinese and Korean, require two bytes to store one character (as against the usual norm of one byte per character). These languages thus, sometimes require UTF-16 encoding that allows storing these extended character sets.

Unicode is an industry standard that supports about 128,000 different characters. It is split into different code point ranges and each each range is mapped to a character range. We describe the relevant code points for emojis in the following section. However, we mention Unicode here to make a point that every character (using any character set) is mapped to a unique unicode code. So you may encounter the terms unicode, UTF-8 and UTF-16 interchangeably in relevant documentation and should learn more to understand the nuances.

Finally, in case you start seeing garbled characters when you try and read your tweets, check your encoding!

### ***Emoticons, Emojis and Kaomoji***

Even though they are used for similar purposes to express different emotions; emoticons, emojis and kaomojis use different character sets.

#### **Emoticons**

Emoticons are predominantly represented using punctuation, letters and numbers. Thus, in most scenarios the ASCII character set is sufficient to express all emoticons. However, a given emoticon may have several variants ( :), :-), :-] are all smiley faces for example, le). Further, the overloaded usage of common characters makes it hard to programmatically distinguish between punctuation usage and emoticons. For example, a regular expression trying to match contiguous punctuation would match both '!!!!' and ':)'. Using a curated lexicon may provide a decent amount of precision but may suffer in terms of recall.

### Kaomojis

There is an alternate “Japanese” style of emoticons ((-\_-;) and (ノ◕□◕) ノゝー for example) that does not require tilting one’s head to understand the emoticon. They use extended character sets, may or may not be enclosed within parenthesis and again, could have multiple variants. Culturally, they may be more frequent in some languages (Korean, Japanese) than others (Turkish, English).

### Emojis

Emojis like , and on the other hand are more expressive ideograms that instead use distinct character sets. Unicode 9.0 reserves 1,123 characters spread across 22 code blocks as emojis. Recently, applicable emojis may be annotated with Fitzpatrick modifiers to indicate diversity ( to for example). One of the decisions that you may have to make is to figure out if you would use the Fitzpatrick modifiers in your indexing process (i.e preserve the modifications) or ignore them completely and only store the default emoji.

Emojis may appear differently between different operating systems. However, all of them map to the same underlying unicode character. Thus, although emojis have fixed unicode ranges, they are slightly more challenging to handle programmatically. You are encouraged to look at specific examples of handling emojis in the programming language you intend to use.