A Framework for Twitter Information Retrieval

Luiz D. R. França[1]

[1]Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco (UFRPE) – Recife, Pe – Brazil

luizdaniel.r.f@gmail.com

Abstract. *With the increasing number of social networks, blogs and personal webpages, the amount of information being uploaded is increasing on a daily basis. It is important to have ways efficient ways of retrieve relevant information to the user. This paper proposes a framework for information retrieval from twitter.*

1. Introduction

The Web 2.0 has changed the way information people consume and produce information. It has allowed them to exchange information in faster ways than ever before. One of the many tools people use to share those ideas are social networks like Twitter.

Twitter is a social network that allows people to exchange 140-character messages. Nowadays many people use twitter to share their ideas and opinions, even companies use it to show their products and interact with their costumers. With the growth of these social networks, a huge amount of data is produced every instant. According to [Internet Live Stats] five hundreds million twitters are uploaded everyday. With so much information being uploaded, it's very hard to get the most relevant twittes (the messages sent on Twitter are called twittes) and extract information from it. So it becomes unfeasible to extract information manually [Ferreira et al 2013]. Messages full of abbreviations and slang and small length of the messages on Twitter makes it even harder to extract relevant information using traditional methods [Sriram et al. 2010]. The information retrieved from Twitter has many applications like predict stock market [Bollen et al. 2011] and monitor political sentiment [Bermingham and Smeaton 2011]. This project will propose an information retrieval framework to index the data extracted from Twitter. This project is relevant because it will make easier to explore the richness of information on Twitter. Furthermore, it's going to allow the retrieval of relevant information that otherwise would be very difficult to find due to the vastness of the contents uploaded everyday on Twitter.

2. Basic Concepts

To better understand the framework, first it's need to understand what is information retrieval, and how does it work.

2.1. Crawlers

Crawlers are programs that visit webpages, adding them to a list of pages while it looks for new links on the pages that could direct to another page. That way, starting from one page the crawler can visit many pages. These pages added are later feed to a search engine to index them.

2.2. Lexical analysis

In text mining, lexical analysis is the first step to create an index. It consists of taking a document and breaking into tokens, so then these words can be processed (see Table 1).

Table 1. Example of lexical analysis

Document	Tokens
Today I'm going to eat cookies	today/I/am/going/to/eat/cookies

2.3. Remove Stop Words

On normal text, there are many words that don't carry much meaning because they occur very often (e.g. the, a, an). These words are removed from the text, so to reduce the amount of unnecessary work and space.

Table 2. Example of removed stop words

Document splitted in tokens	After removing the stop words
today/I/am/going/to/eat/cookies	today/going/eat/cookies

2.5. Lemmatisation

Lemmatisation is the process of reducing the word back to its lemma, therefore changing the word to its inflected form. This is useful avoid the same word have many entries on the index.

Table 3. Example of lemmatisation

Document without the stop words	Lemmas
/today/going/eat/cookies	/today/go/eat/cookie

2.6. Controlled Vocabulary

This phase of the index is responsible for setting the vocabulary that will be used to index the documents (e.g. the set of words the will be used for the inverted index). In this process, the goal is to look for the words that presents most meaning for the documents, store it's frequency and position.

2.7. Inverted index

A inverted index is a map where the key are the terms (or the words) and the value are the documents where this term appears, as well as the positions on the document where the words appear and the frequency (see Table 4).

Table 4. Example of inverted index

Term	Documents
Today	A(0.4, [0, 2, 4]), B (0.3, [1, 4])
Go	A(0.1, [1]),
Cookie	B(0.2, [0, 2])

4. The Framework

The framework is divided into 5 subprocess. The first is the twitter crawler, the second is the store data, third indexing, and fourth search (Figure 1).

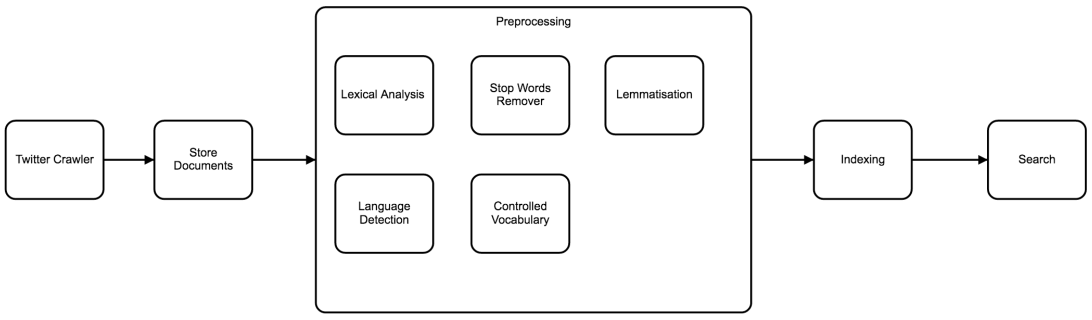


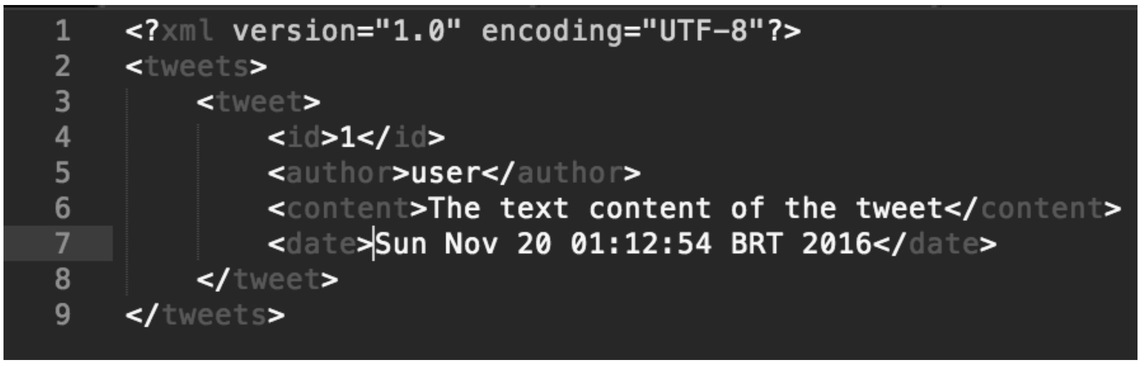
Figure 1. Architecture of the framework

4.1. Twitter Crawler

The twitter crawler is the module responsible for acquiring the tweets. This module works by receiving a initial user as a parameter and then it retrieves the first 3200 tweets (limitation of Twitter API) and goes on each tweet looking for mentions for other users to add to the list of the users to visit. This module also has a parameter for the number of users it should go through, without this limit the crawler would go on forever until it reach the API limit.

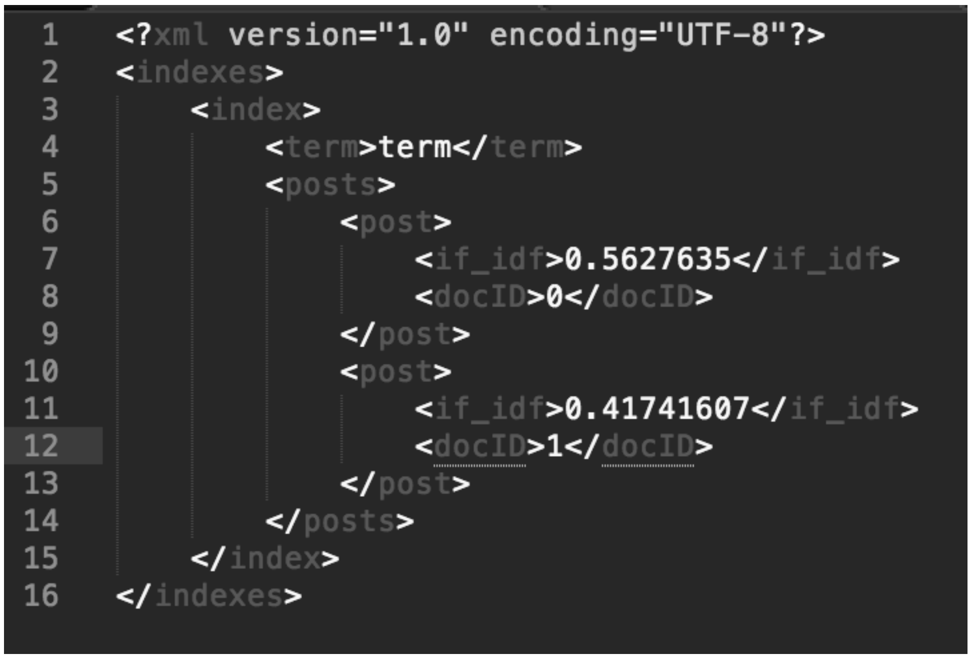
4.2. Store Documents

This module saves the data acquired on the previous step on a XML file structured as shown on the Figure 2. This module also has a function to load the data. The store documents module saves and loads the indexes as well (see Figure 3).



```xml
<?xml version="1.0" encoding="UTF-8"?>
<tweets>
    <tweet>
        <id>1</id>
        <author>user</author>
        <content>The text content of the tweet</content>
        <date>Sun Nov 20 01:12:54 BRT 2016</date>
    </tweet>
</tweets>
```

Figure 2. XML structure of the tweets



```xml
<?xml version="1.0" encoding="UTF-8"?>
<indexes>
    <index>
        <term>term</term>
        <posts>
            <post>
                <if_idf>0.5627635</if_idf>
                <docID>0</docID>
            </post>
            <post>
                <if_idf>0.41741607</if_idf>
                <docID>1</docID>
            </post>
        </posts>
    </index>
</indexes>
```

Figure 3. XML structure of the indexes

4.3. Preprocessing

This module deals with the preprocessing of the data acquired on the step 1. The preprocessing has five substeps as follows:

4.3.1. Lexical Analysis

This module is responsible for tokenizing the texts. This is step is very important, as it serves as an input to all the following steps.

4.3.2. Stop Word Remover

The Stop word remover is responsible for removing all the words that doesn't carry much meaning. It does that by screening the tokens on a list of stop words and removing the word that match the list. It is also removed the punctuations and numbers.

4.3.3. Lemmatisation

This modulo takes as parameter the output from the stop word remover and returns the tokens to their inflected form and it serves as input to the Controlled vocabulary module. On this module I used the CoreNLP pipeline [Manning, Christopher D., et al 2004]

4.3.4. Language Detection

The language detection is handled by the Twitter4j framework [Yamamoto 2007]. As this is the same framework as the one used to retrieve the twitter status, it already comes with the information with respect to the language of the twitter.

4.3.5. Controlled Vocabulary

This module takes in the output from the lemmatization phase and returns a list of the vocabulary list for the indexing as well as the idf score for each term.

4.4. Indexing

The indexing module is responsible for taking the vocabulary list and the list of tweets and produces the inverted index. On these indexes is the information of the tf-idf score of each term with respect to each document.

4.5. Search

The search module takes as parameter a query for search and returns a list of tweets. It does that by taking the query through all phases of preprocessing (lexical analysis, stop word remover and lemmatization) so as to used only the words that carry some meaning and to avoid search for terms that wouldn't be on the index (e.g. the, I, an, and). Then it takes all documents that at least one of the terms of the query and sorts them. This sorting is made by adding the tf idf score of each document (adding all the scores of the terms from the query found on the document). Then this ordered list is returned to the user.

7. Conclusion

Bibliographic references must be unambiguous and uniform. We recommend giving the author names references in brackets, e.g. [Knuth 1984], [Boulic and Renault 1991]; or dates in parentheses, e.g. Knuth (1984), Smith and Jones (1999).

8. Future Work

For the future work, we would like to add the process of text expansion to the preprocessing phase. As the tweets are really short (at most 140-characters long) the amount of information taken from each twitter is little, which makes it harder to index properly.

Also, it would be ideal to add some module to handle the acronyms, slangs and misspelled word of the text, as it is widely used on tweets due to its shortness in length. We believe that this would improve greatly the indexing process, as it wouldn't have many entries for the same term in different formants.

The creation of a new twitter crawler that get the newest twitter statuses for the hashtags from inputted by the user. With this, the user could extract relative and new information from the subjects desired.

References

Internet Live Stats. “Twitter Usage Statistics”, http://www.internetlivestats.com/twitter-statistics/, May 16, 2016

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. (2014). The Stanford CoreNLP Natural Language Processing Toolkit In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55-60.

Bermingham, A. and Smeaton, A. F. (2011). On using twitter to monitor political sentiment and predict election results. *Workshop at the International Joint Conference for Natural Language Processing*.

Bollen, J., Mao, H., and Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*.

Ferreira, R., Freitas, F., Brito, P., Melo, J., Lima, R., and Costa, E. (2013). Retriblog: An architecture-centered framework for developing blog crawlers. *Expert Systems with Application*.

Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., and Demirbas, M. (2010). Short text classification in twitter to improve information filtering. *ACM*.