Consistent topics over time inside tweets

Data Mining Course Project Report

Davide Piva
2nd year CS Master Degree @ University of Trento
davide.piva-1@studenti.unitn.it

1 INTRODUCTION AND MOTIVATION

Twitter, as any other social network and blog, is a constant source of unstructured data that, if processed in the right way, can be leveraged to obtain valuable insights in several fields. Due to this reason, lots of companies have started to collect this huge amount of data in order to perform in-depth analysis on it; many of them have initiated to perform sentiment analysis to check customers satisfaction with respect to their products. Other organizations have started to use this asset, hereafter called Big Data, in their decision-making process that includes reporting, exploration of data and exploratory search (e.g., finding correlations). According to upGrad [12], Big Data can help create pioneering breakthroughs for companies that know how to use it correctly.

A challenging aspect in Big Data analysis is to extract valuable insights that can drive business strategies and decisions. In this context, the social network Twitter plays a crucial role during the information retrieval phase: millions of people publish everyday short messages, hereafter tweets, that often contain their desires or demands. The aim of many companies is to leverage these tweets in order to catch common issues or demands to guide their business strategies for their products or services. These insights, according to upGrad [12], can be used to develop new products/services, enhance marketing techniques, optimize customer service, improve employee productivity and find radical ways to expand brand outreach.

In addition, organizations aim also to intercept occasional issues or demands that are not constant over time but are very popular in a bounded period of time. This is the case when there is a sudden issue or request for a particular product or service that affects many customers. Catching this kind of insights, permits companies to better address their business short-term decision-making phase and enhance promptly their products or services. According to Ben Ridler [3], a business owner ability to effectively deal with customer complaints provides a great opportunity to turn dissatisfied customers into active promoters of the business.

Nowadays, according to [11], there are more than 500 millions new tweets each day and this statistic is still growing every year. Thanks to this fact, many people have started to collect and group those tweets into datasets; many of them are easily and freely accessible by anyone through internet. Despite the fact that many of those datasets contain noisy data that force people to pre-process them, companies can elaborate those tweets in order to achieve the above mentioned business goals.

The aim of this project is to address in an efficient and effective manner the procedure of finding consistent topics over time inside tweet texts. In other words, the goal is to find topics that are frequent locally in a specific time frame but those topics, in order to be considered, must be frequent locally in a sufficient number of time frames. In this report are described in detail the solution to the just mentioned problem and the relative implementation. Finally, in Section 7, is presented an experimental evaluation of this project using the dataset described in Section 3 and 6 [2].

2 RELATED WORK

In order to design and implement this project, several methods and techniques have been identified and implemented; anyway, there are some of them that have been implemented and widely tested but then discarded since they have not produced acceptable results. In the first part of this section are described all those techniques used in order to pre-process the dataset, while in the second part are described all those methods used to find consistent topics over time.

During the pre-processing phase of the dataset have been adopted several techniques used in natural language processing scenarios. The methods that have been implemented and kept in the final preprocessor script are described in details below and applied following this ordering:

- **Sentence segmentation**: the text string of each tweet is divided into sentences. In other words, the result of this step is a list composed by all the sentences written inside the tweet;
- **Tokenization**: with this technique, each sentence obtained from the previous step is split into even smaller parts called *tokens*. In particular, has been used the blank character as string separator: due to this reason, a token corresponds to a word of the sentence;
- **POS Tagging**: this is the most complex task during the pre-processing of the dataset. The idea of the POS (*Part Of Speech*) technique is to assign to each token the corresponding tag (noun, verbs, adjectives, conjunction, etc.). The collection of tags used (tag set) can be considered as standard since it is used widely in the natural language processing community. In order to implement this technique, the map provided by the Python nltk library [8] has been used.

The other technique used during the pre-processing of the dataset is an entity resolution method that is inspired to *lemmatisation* [5]. The just mentioned technique aims to reduce the inflected form of a word in its canonical form in order to group together different words that have the same canonical form. The process implemented in this project is a simpler and non-automatic version of lemmatisation but still very effective on the given dataset. The goal is to define a set of **aliases** stored on the disk in a CSV file interpreted as a dictionary: if a word has an entry in that data structure, it will be replaced with its alias that represents a more general word (see Section 6 for further details).
Another technique that was initially implemented is **stemming** [5]. The aim of this method is to reduce the inflected form of a word in its root form (or "word stem") in order to group together words that have the same root form. The idea was to substitute the given word with its stem during the pre-processing phase. The major problem arose when the script to find the consistent topics over time has been run: the obtained results were not so meaningful since the root form of a word cannot be presented as a topic to the final user. Due to this reason and after many

attempt to improve this approach, the just described technique has been abandoned.

In order to find out all the consistent topics over time, the major technique that has been used is the **frequent itemsets** mining method. The goal of this technique is to identify all the sets of items (in the context of this project, set of words) that appears together a sufficient number of times. In order to achieve the goal of this project, the association rules are not computed since they do not produce any valuable information. In a first attempt, the **A-Priori** algorithm [1] was used and implemented. The goal of this procedure is to find out initially frequent individual terms in the dataset and then extending the scope to larger item sets as long as those item sets appear sufficiently often in the database. The A-Priori algorithm leverages the contra-positive *monotonicity* principle: if a word *s* does not appear in *N* sets, then no pair that includes *s* can appear in *N* sets.

A further attempt was to implement the PCY algorithm or its multihash variation [7], but a more efficient technique that can be parallelized between many machines has been found: **FP-Growth** [4]. The first step of this procedure, similarly to the A-Priori one, is to compute item frequencies and identify frequent items. Then, the second step of FP-Growth is to use a suffix tree structure, called FP-tree, to encode transactions without generating candidate sets explicitly, due to the fact that they are usually expensive to generate. When this second phase terminates the execution, the computed frequent itemsets can be extracted from the FP-tree. With the aim of speed-up the computation distributing the workload among different machines, a parallelizable version of FP-Growth has been used: **Parallel FP-Growth** [6]. This algorithm distributes the work of growing FP-trees based on the suffixes of transactions. Due to this improvement, it is more scalable than a single-machine computation.

The last technique used with the aim to improve the final result presented to the user is a variation of the **closed itemsets** method. The just mentioned technique must be executed after having identified all the frequent itemsets (when the PFP-Growth algorithm has terminated its execution) and its goal is to compact the final output. The idea of closed itemsets is that if there are two frequent itemsets where one is a subset of the other and they have the same frequency, the subset must be discarded. The method implemented in this project follows the same principle, but given two itemsets where one is a subset of the other, the subset is discarded only if it appears in as many time frames as the super set. If they do not appear in the same number of time frames, it means that probably they are different topics so they must be both kept in the result set. The frequency of appearance of the itemsets is not used as a discriminant parameter in order to decide whether to keep or not a subset of items due to the fact that, in any case, a topic composed by many items is much more meaningful than a topic with less items to the final user.

3 PROBLEM STATEMENT

In order to achieve the goal described in Section 1 (i.e., identify consistent topics over time inside tweets), a public available dataset that groups more than 300.000 tweets that contain the hashtag *#covid19* [2] has been used. As many other datasets composed of unstructured data dumped from a social network such as Twitter, this dataset has lots of fields that characterize each tweet (e.g., publisher's username, location and account information, text of the message, etc.).

To identify consistent topics over time using those tweets, the only fields needed are the publication date of each of them and the relative text. In other words, in the working environment a tweet is defined as a tuple composed by the following fields:

- **date**: the date of the tweet publication, expressed as an integer value that represents the number of seconds that have elapsed since the midnight of 1st January 1970;
- **text**: the text of the tweet represented as a list of words. This list is obtained splitting the source text using the blank character " " as separator and processed as described in detail in Section 6.

Therefore, in order to obtain the formal model of this problem, the following sets are been defined:

TWEET = the set of tweets, defined as TIMESTAMP$\times$TEXT where TIMESTAMP $\subset \mathbb{N}_{\geq 0}$ and TEXT $= \{x | x \in \Sigma^*\}$ where Σ is the reference alphabet;

TIMESPAN = the set of all the possible time spans in which the input dataset can be split, defined as a subset of $\mathbb{N}_{\geq 0}$

TS-THRESHOLD = the set of all the possible thresholds to identify frequent terms and topics in a single time span, defined as a subset of $\mathbb{N}_{\geq 0}$

GB-THRESHOLD = the set of all the possible thresholds to identify consistent topics over all the time spans, defined as a subset of $\mathbb{N}$

The aim is to model an utility function f defined as

$$f : \text{INPUT} \mapsto \text{TOPIC}$$

where INPUT is defined as

$$\text{TWEET} \times \text{TIMESPAN} \times \text{TSTHRESHOLD} \times \text{GBTHRESHOLD}$$

and TOPIC $= \{x | x \in \Sigma^*\}$ where Σ is the reference alphabet. In other words, TOPIC is the set of all the possible consistent topics over all the identified time spans.

4 SOLUTION

5 IMPLEMENTATION

6 DATASET

As described previously in Section 3, in order to build and test the solution presented in Section 4 a public available CSV dataset with more than 300.000 tweets that contain the hashtag *#covid19* [2] has been downloaded from kaggle.com and processed. As many other datasets that group together huge quantity of unstructured data, it had to undergo a pre-processing phase in order to remove unnecessary features and noisy data.

In that dataset, a tweet is defined as a data item composed of thirteen features: *user-name*, *user-location*, *user-description*, *user-created*, *user-followers*, *users-friends*, *user-favourites*, *user-verified*, *date*, *text*, *hashtags*, *source*, *is-retweet*. In order to achieve the goal described in Section 3, only two features are needed: the publication date of the tweet and the relative text. Due to this reason, the first stage of the pre-processing consists to cut off all the non-relevant data features. This has been achieved quite easily since the dataset is CSV (Comma Separated Value) file and the Pandas Python library [9] has lots of APIs to manage efficiently such big datasets.

After this initial stage, each tweet has been processed singularly in a multi-process execution where to each process is assigned a portion of the input dataset. For each tweet in the dataset portion, each process has to manipulate both the date and the text fields in the following manner:

Date field is transformed into an UNIX timestamp. In particular the publication date is stored in the input dataset as a string formatted as "yyyy-mm-dd hh:mm:ss", but thanks to the *datetime* Python library this transformation can be performed in a easy and fast way. With the aim to be complaint also with the format of another dataset that groups together tweets in order to perform sentiment analysis [10], the script responsible for the date manipulation is capable to transform also dates formatted like "Mon Apr 06 22:19:45 PDT 2009" into a UNIX timestamp.

Text field is transformed into a list of words. In particular, thanks to the *nltk* [8] Python library, the initial text has been split using the blank character " " as separator and only the useful words that can be leveraged in order to derive a topic have been kept. Due to this reason, only nouns and adjectives appear in the final result list. Furthermore, there are other situations where a word, even it is a name or an adjective, cannot be considered:

- if the word contains a slash "/", non-ASCII characters, special Unicode sequences;
- if the word represents a numeric value;
- if the word is composed by only one character;
- if the word is a stop word. This method has been implemented because there are some words like "https" or "amp" that are tagged as nouns but they cannot be leveraged to build a topic.

As a note, if the preprocessor script has been executed in the debug mode, all the non-considerable words are dumped inside a separated CSV file (together with the relative timestamp) in order to check if the script cut off useful words. Then, each considerable word is filtered and sanitized in order to remove all the eventual noisy characters. In particular, the filtering method performs the following operations:

(1) all the word's characters are lowered;
(2) all the emojis are removed;
(3) all the non-alphanumeric characters are substituted with a blank space (e.g., the word "white-house" is transformed to "white house");
(4) the word is split again using the blank character as separator to identify eventual other words after the previous operation;
(5) all the obtained words are rechecked again to see if they are considerable and, eventually, discarded;
(6) as a final operation, each of the obtained words are checked with an aliases map. If there is an entry for a word, then the associated value is substituted. The aim is to generalize as much as possible the words used inside the tweets leveraging the knowledge about that dataset in order to obtain better results: since all the texts contain the word "covid19" or one of its many variations (e.g., "sars-cov-2", "covid", "coronavirus", etc.), all of them are mapped to the word "covid19".

After these filtering operations, a final check is performed over all the considerable words in order to find out if there are doubled words or empty string.

When each process has terminated its execution, all the partial results are collected in a single list by the master process using a pipe-based communication and sort that list in ascending order using the timestamp field as key. The result is a list of tweets ordered by their publication: this feature can be leveraged by the main script described in Section 5 in order to lower the overall complexity. If the preprocessor script has been executed in debug mode, also the list that contains all the discarded words would be collected but not sorted.

As a final step, the result list has been converted by the master process in a Pandas [9] dataframe object and stored in a separated CSV file. The same procedure would be applied to the second list with all the discarded words if the preprocessor script has been executed in debug mode.

7 EXPERIMENTAL EVALUATION

7.1 Test dataset

REFERENCES

[1] apriori-algorithm [n.d.]. *A-Priori algorithm reference*. Retrieved January 10, 2021 from https://en.wikipedia.org/wiki/Apriori_algorithm

[2] Covid19-Dataset [n.d.]. *Tweets with the hashtag covid19*. Retrieved December 17, 2020 from https://www.kaggle.com/gpreda/covid19-tweets

[3] Customer-Complaints [n.d.]. *Six Steps to Dealing with Customer Complaints*. Retrieved December 18, 2020 from https://www.eonetwork.org/octane-magazine/special-features/sixstepstodealingwithcustomercomplaints

[4] Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining Frequent Patterns without Candidate Generation. *SIGMOD Rec.* 29, 2 (May 2000), 1–12. https://doi.org/10.1145/335191.335372

[5] lemmatisation-stemming [n.d.]. *Lemmatisation and stemming reference*. Retrieved January 10, 2021 from https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

[6] Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang, and Edward Y. Chang. 2008. Pfp: Parallel Fp-Growth for Query Recommendation. In *Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys '08)*. Association for Computing Machinery, New York, NY, USA, 107–114. https://doi.org/10.1145/1454008.1454027

[7] pcy-multihash-algorithms [n.d.]. *PCY and multihash algorithms reference*. Retrieved January 10, 2021 from https://medium.com/weekly-data-science/the-pcy-algorithm-and-its-friends-ecba67216190

[8] Python-nltk [n.d.]. *Natural Language ToolKit python library*. Retrieved January 10, 2021 from https://www.nltk.org/

[9] Python-Pandas [n.d.]. *Pandas python library*. Retrieved January 10, 2021 from https://pandas.pydata.org/

[10] Sentiment-Analysis-Dataset [n.d.]. *Tweets for sentiment analysis*. Retrieved January 10, 2021 from https://www.kaggle.com/kazanova/sentiment140

[11] Twitter-statistics [n.d.]. *Twitter usage statistics*. Retrieved January 10, 2021 from https://www.internetlivestats.com/twitter-statistics/

[12] upGrad [n.d.]. *Benefits and Advantages of Big Data and Analytics in Business*. Retrieved December 17, 2020 from https://www.upgrad.com/blog/benefits-and-advantages-of-big-data-analytics-in-business/