Web Information Retrieval

Earthquake Shakes *Twitter* Users: Event Detection by Social Sensors

Matteo Mariani
Matricola: 1815188

Riccardo Chiaretti
Matricola: 1661390

Federico Mascoma
Matricola: 1653309

ABSTRACT

Microblogging platforms such as Twitter provide active communication channels during crisis or calamity events such as earthquakes, typhoons or conflicts. During these events, affected people post useful information on Twitter that can be used for situational awareness and other humanitarian disaster responses, if processed timely and effectively.
The examined paper [1] investigates real-time detection of typhoons and earthquakes through tweet's analysis. To detect a target event, they devised a classifier of tweets based on features such as the presence of keywords in a tweet, the number of words and their context. After tweets' classification, they produced a probabilistic spatio-temporal model for the target event that tries to guess both the center and the trajectory of the event and the time at which it stroke.
For spatial detection, they considered each Twitter users as a sensor applying some techniques like Kalman filtering[1] and particle filtering [2] which are widely used for estimating the location.
For time detection, they implement a burst detection method based on probability distributions and heuristics. They designed a model around the exponential distribution, introducing and tuning parameters in order to correctly detect the exact time of a strike.
Lastly, they implemented a real-time earthquakes notification system exploiting the classifier previously trained to recognize crisis events and send, promptly, an email to all registered and affected users once the system detected an earthquake.

Authors Keywords

Event detection; Twitter; CrisisNLP dataset; stemming; lemmatization; semantic analysis; Out of Vocabulary; feature extraction; Word2vec embeddings; support vector machine; k-nearest neighbors; spatio-temporal model.

[1]Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone. [2]

[2]A particle filter is a probabilistic approximation algorithm implementing a Bayes filter, and a member of sequential Monte Carlo methods. [3]

1. INTRODUCTION

Inspired by the previously described paper, we designed a tweet-based earthquake detection system applying some modifications and simplification. In this section, we will briefly describe the guidelines we have used for the development of our project. We will explain them more in detail in the following sections.

The authors developed a real-time earthquake reporting-system using Japanese tweets. Due to the huge number of earthquakes in Japan and the high number of Twitter users throughout the country, they could detect earthquakes by monitoring directly the tweets. Since this dataset was not publicly available and due to time limitations and constraints imposed by Twitter social platform regarding retrieving information from it, we could not follow the same approach. Moreover, even if Japan is one of the country that suffers the most from earthquakes and has an high number of Twitter users, we could not use Japanese tweets because of its particular language syntax.[3]
To overcome these issues, we used a dataset which will be described in Section 2.

After having prepared the training data through some preprocessing technique (Section 4) and having selected the types of features to extract from tweets (Section 5) we devised a binary classifier using *Support Vector Machine* (SVM) - as the authors suggested - and, in addition, we also used *K-Nearest Neighbors* (kNN) in order to make some interesting comparisons (Section 6).

Subsequently, the trained classifier has been used for designing the spatio-temporal model. At this point, we made some simplifications with respect to the assigned paper (Section 7).
For the temporal model we implemented a burst detection technique which uses the posting time of each crisis-related tweet to obtain an approximation of the correct strike time of an earthquake.
For the spatial model, instead, we exploited geo-tagged crisis-related tweets in order to determine approximately the epicenter of an earthquake through latitude and longitude information, when available.[4]

[3]This detail would become a limitation in the preprocessing phase since some techniques would have been more difficult to apply.

[4]Note that, since the detection is made using users' position, we can not estimate exactly the real epicenter of an earthquake but we tried to reach a good approximation.

Lastly, we were not interested in the development of a real-time earthquake notification system since this was unpractical and far beyond the project's scope.

2. *CRISIS NLP DATASET*

Availability of data - in particular human-annotated one - is one of the major problem that may occur in the development of a learning system.

Since the authors of the paper did not share any dataset, we have decided to use the well-known *CrisisNLP* dataset. [4]
This dataset is composed of millions of unlabeled tweets related to 19 different crises that took place between 2013 and 2015, with roughly 20.000 labeled tweets. These tweets are composed simply by the tweet ID and the user ID and they were collected in different languages.

The ground truth is composed of human-annotated data that not only identifies a tweet as crisis-related or not, but also classifies tweets in 8 sub-classes related to - for example - "infrastructure damage", "missing people" or "evacuations problems". For each labeled tweet we have also some important information like the tweet ID, the tweet's author, the posting date and time and the tweet message.

Using this dataset, we were able to use two important tools that improved in different ways our project with respect to the original paper. Those tools are, respectively, *Word2vec embeddings* and an *Out of Vocabulary* (OoV).

- CrisisNLP dataset offers a ready-to-use Word2vec embeddings trained using crisis related tweets. Word2vec is a group of related models that are used to produce word embeddings.[5] It takes as input a large corpus of text and produces a vector space, typically of several hundred dimensions, where, each word, corresponds to a vector in that space. Word vectors are placed in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.

- an Out of Vocabulary is a kind of dictionary used in computational linguistics and natural language processing in which, for each unrecognized term in a given vocabulary, there is the correspondent correct term in that vocabulary.

The Word2vec will be used to improve one of the feature extraction methods suggested by the original paper (Section 5), instead the Out of Vocabulary will be fundamental in the preprocessing phase (Section 4).

We made some modifications to the dataset in order to adapt it to our needs. First, we focused only on earthquakes related tweets without considering any other kind of crisis like typhoons, floods and so on. Secondly, we collapsed the 9 classes into 2 classes (crisis-related tweet or not), considering a binary classification problem. Lastly, we focused only on English tweets since we were able to use the Out of Vocalubary based on English words.

[5]Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers.

3. RETRIEVING TWEETS

As we previously mentioned, the dataset used in the project is totally different from the one used by the paper's authors. In addition, even if we have found a dataset which was effectively useful for our purposes, it did not have all the information we needed.
Indeed, for implementing the spatio-temporal model, we needed more information related to position of users and tweets.

For this purpose, we used a completely open-source Python library: `Tweepy` [5]. Using this, we were able to retrieve all necessary data related both to the location in which the tweet was posted and the position registered in the user account. Yet, there were some problems: Twitter makes possible to do only 900 requests each 15 minutes and not all tweets could have been retrieved due to (1) canceled or private accounts or (2) removed tweets.
At the end we obtained, for each earthquake, a file containing all the needed information.

4. PREPROCESSING

When retrieving tweets using the API offered by the social platform, the message could also include URLs, digits and special characters like hash symbols or also "at" signs. Clearly, that is not necessary for classification purposes, actually it can lead to biased results.
For this reason, we applied some standard pre-processing methods like punctuation and stop words elimination, URL deletion, symbols and digits elimination.

Moreover, since Twitter is a micro-blogging platform, users often tend to use slang words or also acronyms for particular expressions like "Oh my god" in OMG or also BTW for "By the way". In addition, some concept can be expressed using different words like *earthquake* or *earthquakes*, *shake* or *shaking*; users can even use capital letters within a word. As a consequence, in order to reduce the size of the vocabulary, we applied some pre-processing techniques using `nltk` library like:

- *stemming* that is an heuristic which simply chops off the end of a word;

- and *lemmatization* which, on the other hand, uses a vocabulary and morphological analysis of a word to return the base form (also known as *lemma*) of the word itself.

In that way, we have solved the problem of having different words for expressing one concept. Yet, we needed to find a way for expanding acronyms and translating slangs to their English version. This task is achieved using the *Out-of-Vocabulary*: this file contains, for each misspelling, slang and acronym, the correspondent English word or sentence.

5. FEATURE EXTRACTION

Before starting the binary classification of tweets, we analyzed which kind of features to extract from each tweet. As the paper suggested, some possible features to extract can be:

- **Feature A** (statistical feature): the number of words and the position of the query word within a tweet.

- **Feature B** (keyword features): the occurrence of words in a tweet.
- **Feature C** (word context features): the words before and after the query word.

The query words are essentially a set of key words - stemmed or lemmatized, it depends from which kind of preprocessing mode we are considering when doing feature extraction - that are strongly related to earthquake disaster.[6]

Now, let's analyze each feature extraction method.
Note that each of them will produce two results: a feature matrix and a label vector. The feature matrix has a number of rows that is independent on the type of feature that we're extracting - since it represents the number of tweets in the dataset - and a number of columns that is strictly dependent on the feature. The label vector is a one-hot vector containing a 1 if the i-th tweet is crisis-related, 0 otherwise; it is clearly independent from the feature extraction method.

5.1 Feature A - Statistical feature

Feature A considers the number of words in a tweet and the position of the first query word within a tweet. [7]

This was the easiest feature to extract: each tweet was represented as a couple of integers where the first element is the message length and the second term is the position of the first query word in the tweet, if present; -1 otherwise.

Regardless its simplicity, this feature was the most problematic to treat in the classification phase due to the low dimensionality of the feature matrix, leading to high missclassification from classifiers like linear SVM that performs badly on low dimensional spaces. We will see how we found a solution to this problem when describing the classification methods in the Section 8.

5.2 Feature B - Keyword features

Feature B considers the occurrences of words within a tweet.

Therefore, we are talking about a count matrix: it is a document-term matrix having as columns the terms in the corpus. Since the obtained matrix would be *very* sparse, it has been implemented as a dictionary containing, for each tweet, only the occurrences of terms appearing in that tweet - it is a compact structure of a sparse matrix.

5.3 Feature C - Word context features

Feature C considers the words before and after the first query word within a tweet.

Here, we used the Word2vec embeddings trained directly with the CrisisNLP.
Given a tweet message with a query word, we first took all the words before and after the query word and then we queried the Word2vec model to retrieve the 300-sized representation vector of each word of the right and left context.
After that, we computed the mean vectors for the entire left context and right context: in this way, we obtained two 300-sized vectors that represented perfectly the context surrounding a query word.
The final step was simply concatenating these two vectors: the result will be one row of the feature matrix.

[6]For example, the words "earthquake" or "shaking" will have more importance in the analysis with respect to others. In our project, `queryWords = ["earthquake","shake","magnitude", "quake","strike","tsunami"]`.

[7]Note that, key words in `query words` are sorted in decreasing order of importance.

6. CLASSIFICATION METHODS

As the authors proposed, we prepared the training data and devised a classifier using Support Vector Machine (SVM) on features previously described. In addition, we decided to use different kernels (*linear* and *radial basis function*) in order to make some comparisons on the different extracted features.

Moreover, we used the K-Nearest Neighbors (kNN) classifier since it suits well to handle low dimension data.

7. SPATIO-TEMPORAL MODEL

Once the model has been trained, this can be used to classify new tweets related to other crisis. Therefore, in the following, we are going to explain how a spatio-temporal model has been devised for estimating the time and the center at which the crisis event has stroke.

7.1 Temporal estimation

Since the authors have implemented a event notification system, they have estimated how much time to wait until sending the notification to all users. For this reason, they followed a probabilistic approach to determine, at some time t, whether a catastrophic event has happened or not.

We decided to follow another way of dealing with the problem. At first, to simulate a real-time arrival of tweets, we sorted them according to their creation time. Then, we grouped tweets in temporal windows of 1 minute each and, with the help of a burst detection library [7], we selected only those windows which are believed to contain a burst of positive tweets.
At the end, by only considering the creation time of those messages, we computed their mean an median having an estimate of the incidence time.

7.2 Spatio estimation

So, after having an estimation of the stroke time of the event, we started collecting the position related to each tweet. Since that not all tweets have been published with a position, we also took the position the user registered in her profile. This is just because, as the authors say, it is reasonable to think that most of tweets are published from users registered in the hit area.
Given information related to the country like "San Francisco, CA", in order to retrieve the correspondent latitude and the longitude, we used the open-source Python library `geopy`. [8]
For those tweets for which we were able to retrieve their coordinates, we computed both their mean and their median. At the end, the best spatial estimation for the center of each earthquake is given by the median of their coordinates.

8. RESULTS

During our experiments we noticed that the performances of our classifiers were negatively affected by the imbalance of the dataset. The bias came from the higher number of positive instances with respect to the negative ones: the negative samples were about 5% of the total. To overcome this issue we considered appropriate to insert some negative samples (roughly 30%) in order to smooth the disparity between classes. Successively, this approach led to a significant improvement.

In order to tune the parameters of the classifiers in the best way, we relayed on a facility offered by `sklearn` library [9]: the `GridSearchCV`. Given a dataset, this performs an exhaustive search over a set of parameters for a given estimator, selecting the best setting.

In the following, we're going to present the results we achieved with classifiers. Then, we will compare the performances between the different feature extraction methods.

Table 1. Classification results.
Compared results between each feature (A, B, C) and each classifier (SVM with linear or rbf kernel and kNN). For each feature we differentiate between stemmed (s) and lemmatized (l) text.

SVM - linear kernel

Feature	Accuracy	Precision	Recall	F1-measure
A (s)	70%	49%	70%	58%
A (l)	70%	49%	70%	58%
B (s)	96%	96%	95%	96%
B (l)	97%	97%	97%	97%
C (s)	92%	91%	92%	91%
C (l)	93%	93%	93%	93%
All (s)	96%	96%	96%	96%

SVM - radial basis function kernel

Feature	Accuracy	Precision	Recall	F1-measure
A (s)	71%	68%	71%	65%
A (l)	71%	69%	72%	67%
B (s)	96%	97%	96%	96%
B (l)	97%	97%	97%	97%
C (s)	92%	92%	92%	92%
C (l)	95%	95%	95%	95%
All (s)	97%	97%	97%	97%

K-Nearest Neighbors

Feature	Accuracy	Precision	Recall	F1-measure
A (s)	67%	68%	78%	68%
A (l)	70%	71%	70%	70%
B (s)	89%	90%	89%	88%
B (l)	91%	92%	91%	90%
C (s)	90%	90%	90%	90%
C (l)	91%	91%	91%	90%

For each combination of feature and classifier, we collected relevant performance metrics described in the table 1.

Regarding feature A, we already introduced the problem related to low dimensionality of the vector space in which the feature lies. In practice, the linear kernel tends to perform very well when the number of features is large - since there is no need to map to an even higher dimensional space. Therefore, as shown by the results, we increased the performances of our SVM classifier by using a more appropriate kernel: the radial basis function. Using k-Nearest Neighbors the performance were boosted even more, reaching outstanding results regardless dimensionality problem. In a low dimensional space we have a better and precise definition of distance between points that blurs when increasing dimensionality. Since kNN classifies points according to their distance with respect to the nearest point, feature A is suitable to the characteristics of the classifier.

Concerning feature B and C, SVM performs well in both cases independently from which kernel is used. Moreover, SVM outperforms kNN; largely.[8]

In general, the results obtained with stemming or lemmatization are almost equal; although the benefit is small, lemmatized text represents corpus of better quality.

After these experiments, we tried to combine all the features together and classify the tweets according to the new feature matrix; as the authors do. Exploiting the best setting given by the previous classification sessions, we trained a new model by using stemmed text and SVM. Since combining together all features led to a huge matrix and considering our limited resources, we decided to do only few experiments, as the table shows.

Regarding the spatio-temporal model; for each earthquake we estimated both the epicenter and strike time, as can be seen in table 2. However, we have to emphasize the fact that, for each earthquake in the dataset, tweets related to a particular earthquake were collected starting the day after the strike. For this reason, as we can see from the table, we were not able to give an estimation which falls into the same day of the event but we tried to smooth the problem by introducing some noise in in a determined temporal window.[9] Quite the opposite is the result we obtained from the spatial model. Although authors of the original paper used more suitable techniques for the estimation of the epicenter, implementing a simple mean and median calculations give us a good approximation of the center.

In particular, the median gives the best result both in temporal and spatial model.

[8] As expected, since we face the opposite scenario with respect to feature A.

[9] For example, if an earthquake stroke on April 25, usually the dataset have tweets related to that earthquake starting from April 26. Because of this problem, we introduce new noisy tweets with a random creation time that span from some days before the earthquakes to some days after the earthquakes. In this way, the input to the temporal model was quite unbiased.

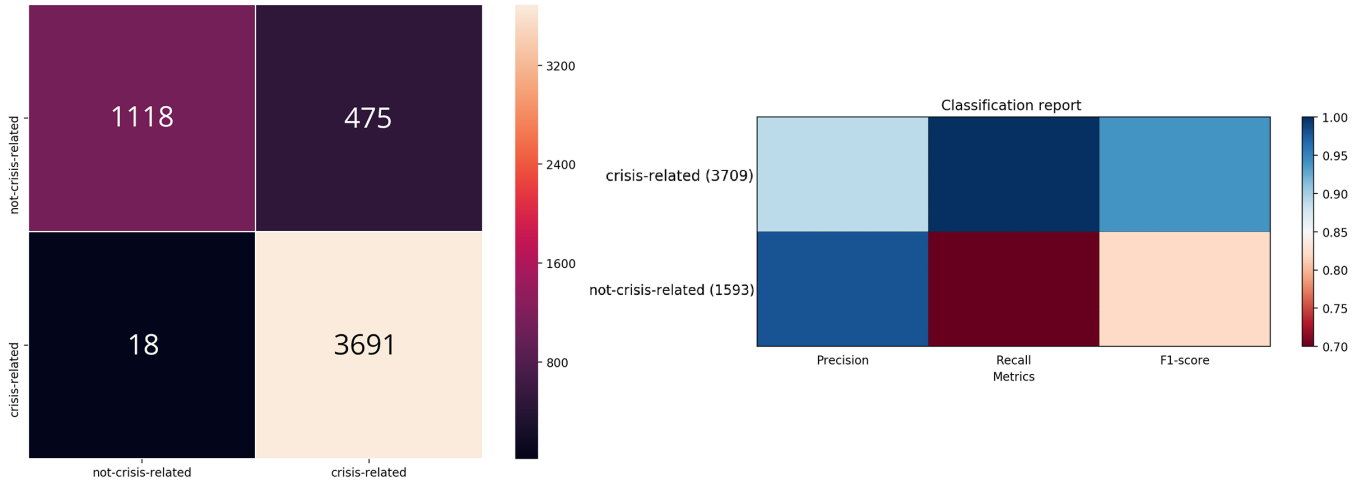

Figure 1. Confusion matrix and classification report

Example of results using linear SVM and stemming technique over feature extraction B. The left-side figure represents the confusion matrix that put on evidence the errors that the classifier did on the validation set (30% of the dataset). The right-side figure represents a graphical classification report with precision, recall and F1-measure.

Table 2. Spatio-temporal model results.

The first table compares the predicted *time* (obtained by the mean or median) and the real time of a strike in different countries. The second table compares the predicted *location* (obtained by the mean or median) and the real epicenter of a strike in different countries. In both analysis and for each earthquake, we highlighted the error in the prediction.

Time prediction

Earthquake	Mean	Median	Real
Pakistan	09-25 11:01	09-25 10:55	09-24 11:29
Chile	04-02 11:17	04-02 02:47	04-01 23:46
Nepal	04-26 13:15	04-26 02:45	04-25 06:11
California	08-24 22:51	08-24 20:45	08-24 10:20

Spatial prediction

Earthquake	Mean	Median	Real
Pakistan	59.2, -0.31	30.33, 67.18	26.97, 65.52
Chile	89.16, 10.32	-35.9 -79.3	-19.6, -70.8
Nepal	70.05, 0.35	28.65, 27.53	28.23, 84.73
California	84.15, 178.1	38.58, -92.83	38.2, -122.3

9. IMPROVEMENTS

In this section, we will briefly describe some possible improvements to our project.

Since the combined extraction of feature A, B and C requires high computational power due to the high dimension of the final feature matrix; this easily represents the first aspect that should be revisited. *Principal Component Analysis* (PCA) is a technique of dimension reduction that could help in treating the problem we have faced, allowing to reduce the computational and cost complexities. In particular, PCA it's used when we need to tackle the curse of dimensionality among data with linear relationship, getting rid of redundancy of data.

Regarding the extraction of feature B, we could implement a different approach: instead of considering just a count matrix, we could use a tf-idf representation of the same matrix. This corresponds to a matrix having a tf-idf weights in place of occurrence values.

10. REFERENCES

1. *Takeshi Sakaki, Makoto Okazaki*, and *Yutaka Matsuo*. 2010. Earthquake shakes Twitter users: real-time event detection by social sensors. ACM, New York, NY, USA, 851-860. DOI=http://dx.doi.org/10.1145/1772690.1772777
2. Wikipedia - Kalman filter, https://en.wikipedia.org/wiki/Kalman_filter
3. Wikipedia - Particle filter, https://en.wikipedia.org/wiki/Particle_filter
4. CrisisNLP dataset, http://crisisnlp.qcri.org/lrec2016/lrec2016.html
5. Tweepy library, http://tweepy.readthedocs.io/en/v3.5.0/
6. Natural Language Toolkit library, https://www.nltk.org/
7. Burst detection library, https://pypi.org/project/burst_detection/
8. Geopy library, https://geopy.readthedocs.io/en/stable/
9. Sklearn library, http://scikit-learn.org/stable/documentation.html