A Support Platform for Event Detection using Social Intelligence

Timothy Baldwin, Paul Cook, Bo Han, Aaron Harwood,
Shanika Karunasekera and **Masud Moshtaghi**
Department of Computing and Information Systems
The University of Melbourne
Victoria 3010, Australia

Abstract

This paper describes a system designed to support event detection over Twitter. The system operates by querying the data stream with a user-specified set of keywords, filtering out non-English messages, and probabilistically geolocating each message. The user can dynamically set a probability threshold over the geolocation predictions, and also the time interval to present data for.

1 Introduction

Social media and micro-blogs have entered the mainstream of society as a means for individuals to stay in touch with friends, for companies to market products and services, and for agencies to make official announcements. The attractions of social media include their reach (either targeted within a social network or broadly across a large user base), ability to selectively publish/filter information (selecting to publish certain information publicly or privately to certain groups, and selecting which users to follow), and real-time nature (information "push" happens immediately at a scale unachievable with, e.g., email). The serendipitous takeoff in mobile devices and widespread support for social media across a range of devices, have been significant contributors to the popularity and utility of social media.

While much of the content on micro-blogs describes personal trivialities, there is also a vein of high-value content ripe for mining. As such, organisations are increasingly targeting micro-blogs for monitoring purposes, whether it is to gauge product acceptance, detect events such as traffic jams, or track complex unfolding events such as natural disasters.

In this work, we present a system intended to support real-time analysis and geolocation of events based on Twitter. Our system consists of the following steps: (1) user selection of keywords for querying Twitter; (2) preprocessing of the returned queries to rapidly filter out messages not in a pre-selected set of languages, and optionally normalise language content; (3) probabilistic geolocation of messages; and (4) rendering of the data on a zoomable map via a purpose-built web interface, with facility for rich user interaction.

Our starting in the development of this system was the Ushahidi platform,[1] which has high uptake for social media surveillance and information dissemination purposes across a range of organisations. The reason for us choosing to implement our own platform was: (a) ease of integration of back-end processing modules; (b) extensibility, e.g. to visualise probabilities of geolocation predictions, and allow for dynamic thresholding; (c) code maintainability; and (d) greater logging facility, to better capture user interactions.

2 Example System Usage

A typical user session begins with the user specifying a disjunctive set of keywords, which are used as the basis for a query to the Twitter Streaming API.[2] Messages which match the query are dynamically rendered on an OpenStreetMap mash-up, indexed based on (grid cell-based) location. When the user clicks on a location marker, they are presented with a pop-up list of messages matching the location. The user can manipulate a time slider to alter the time period over which to present results (e.g. in the last 10 minutes, or over

[1] http://ushahidi.com/

[2] https://dev.twitter.com/docs/streaming-api

Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, pages 69–72,
Avignon, France, April 23 - 27 2012.

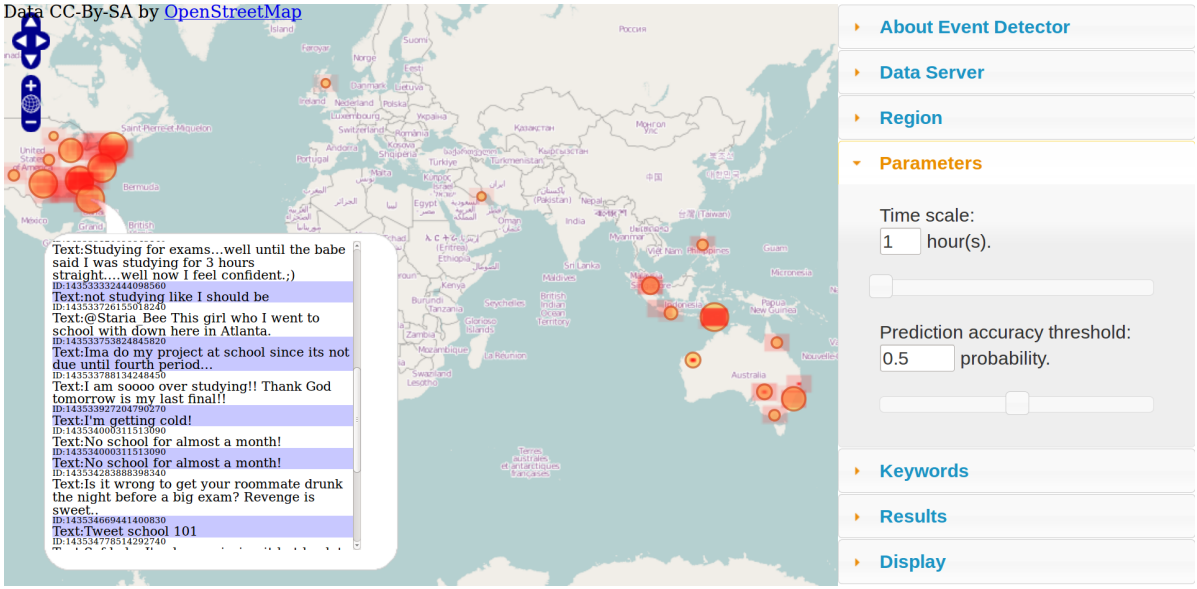

Figure 1: A screenshot of the system, with a pop-up presentation of the messages at the indicated location.

the last hour), to gain a better sense of report recency. The user can further adjust the threshold of the prediction accuracy for the probabilistic message locations to view a smaller number of messages with higher-confidence locations, or more messages that have lower-confidence locations.

A screenshot of the system for the following query is presented in Figure 1:

> *study studying exam "end of semester" examination test tests school exams university pass fail "end of term" snow snowy snowdrift storm blizzard flurry flurries ice icy cold chilly freeze freezing frigid winter*

3 System Details

The system is composed of a front-end, which provides a GUI interface for query parameter input, and a back-end, which computes a result for each query. The front-end submits the query parameters to the back-end via a servlet. Since the result for the query is time-dependent, the back-end regularly re-evaluates the query, generating an up-to-date result at regular intervals. The front-end regularly polls the back-end, via another servlet, for the latest results that match its submitted query. In this way, the front-end and back-end are loosely coupled and asynchronous.

Below, we describe details of the various modules of the system.

3.1 Twitter Querying

When the user inputs a set of keywords, this is issued as a disjunctive query to the Twitter Streaming API, which returns a streamed set of results in JSON format. The results are parsed, and piped through to the language filtering, lexical normalisation, and geolocation modules, and finally stored in a flat file, which the GUI interacts with.

3.2 Language Filtering

For language identification, we use `langid.py`, a language identification toolkit developed at The University of Melbourne (Lui and Baldwin, 2011).[3] `langid.py` combines a naive Bayes classifier with cross-domain feature selection to provide domain-independent language identification. It is available under a FOSS license as a stand-alone module pre-trained over 97 languages. `langid.py` has been developed specifically to be able to keep pace with the speed of messages through the Twitter "garden hose" feed on a single-CPU machine, making it particularly attractive for this project. Additionally, in an in-house evaluation over three separate corpora of Twitter data, we have found `langid.py` to be overall more accurate than other state-of-the-art language identification systems such as

[3] `http://www.csse.unimelb.edu.au/research/lt/resources/langid`

`lang-detect`[4] and the Compact Language Detector (CLD) from the Chrome browser.[5]

`langid.py` returns a monolingual prediction of the language content of a given message, and is used to filter out all non-English tweets.

3.3 Lexical Normalisation

The prevalence of noisy tokens in microblogs (e.g. *yr* "your" and *soooo* "so") potentially hinders the readability of messages. Approaches to lexical normalisation—i.e., replacing noisy tokens by their standard forms in messages (e.g. replacing *yr* with *your*)—could potentially overcome this problem. At present, lexical normalisation is an optional plug-in for post-processing messages.

A further issue related to noisy tokens is that it is possible that a relevant tweet might contain a variant of a query term, but not that query term itself. In future versions of the system we therefore aim to use query expansion to generate noisy versions of query terms to retrieve additional relevant tweets. We subsequently intend to perform lexical normalisation to evaluate the precision of the returned data.

The present lexical normalisation used by our system is the dictionary lookup method of Han and Baldwin (2011) which normalises noisy tokens only when the normalised form is known with high confidence (e.g. *you* for *u*). Ultimately, however, we are interested in performing context-sensitive lexical normalisation, based on a reimplementation of the method of Han and Baldwin (2011). This method will allow us to target a wider variety of noisy tokens such as typos (e.g. *earthquak* "earthquake"), abbreviations (e.g. *lv* "love"), phonetic substitutions (e.g. *b4* "before") and vowel lengthening (e.g. *goooood* "good").

3.4 Geolocation

A vital component of event detection is the determination of where the event is happening, e.g. to make sense of reports of traffic jams or floods. While Twitter supports device-based geotagging of messages, less than 1% of messages have geotags (Cheng et al., 2010). One alternative is to return the user-level registered location as the event location, based on the assumption that most users report on events in their local domicile. However, only about one quarter of users have registered locations (Cheng et al., 2010), and even when there is a registered location, there's no guarantee of its quality. A better solution would appear to be the automatic prediction of the geolocation of the message, along with a probabilistic indication of the prediction quality.[6]

Geolocation prediction is based on the terms used in a given message, based on the assumption that it will contain explicit mentions of local place names (e.g. *London*) or use locally-identifiable language (e.g. *jawn*, which is characteristic of the Philadelphia area). By including a probability with the prediction, we can give the system user control over what level of noise they are prepared to see in the predictions, and hopefully filter out messages where there is insufficient or conflicting geolocating evidence.

We formulate the geolocation prediction problem as a multinomial naive Bayes classification problem, based on its speed and accuracy over the task. Given a message m, the task is to output the most probable location $loc_{max} \in \{loc_i\}_1^n$ for m. User-level classification can be performed based on a similar formulation, by combining the total set of messages from a given user into a single combined message.

Given a message m, the task is to find $\arg\max_i P(loc_i|m)$ where each loc_i is a grid cell on the map. Based on Bayes' theorem and standard assumptions in the naive Bayes formulation, this is transformed into:

$$\arg\max_i P(loc_i) \prod_j^v P(w_j|loc_i)$$

To avoid zero probabilities, we only consider tokens that occur at least twice in the training data, and ignore unseen words. A probability is calculated for the most-probable location by normalising over the scores for each loc_i.

We employ the method of Ritter et al. (2011) to tokenise messages, and use token unigrams as features, including any hashtags, but ignoring twitter mentions, URLs and purely numeric tokens. We

[4] `http://code.google.com/p/language-detection/`

[5] `http://code.google.com/p/chromium-compact-language-detector/`

[6] Alternatively, we could consider a hybrid approach of user- and message-level geolocation prediction, especially for users where we have sufficient training data, which we plan to incorporate into a future version of the system.

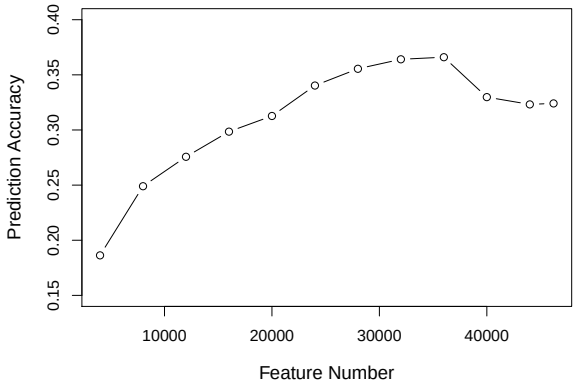

Figure 2: Accuracy of geolocation prediction, for varying numbers of features based on information gain

also experimented with included the named entity predictions of the Ritter et al. (2011) method into our system, but found that it had no impact on predictive accuracy. Finally, we apply feature selection to the data, based on information gain (Yang and Pedersen, 1997).

To evaluate the geolocation prediction module, we use the user-level geolocation dataset of Cheng et al. (2010), based on the lower 48 states of the USA. The user-level accuracy of our method over this dataset, for varying numbers of features based on information gain, can be seen in Figure 2. Based on these results, we select the top 36,000 features in the deployed version of the system.

In the deployed system, the geolocation prediction model is trained over one million geotagged messages collected over a 4 month period from July 2011, resolved to 0.1-degree latitude/longitude grid cells (covering the whole globe, excepting grid locations where there were less than 8 messages). For any geotagged messages in the test data, we preserve the geotag and simply set the probability of the prediction to 1.0.

3.5 System Interface

The final output of the various pre-processing modules is a list of tweets that match the query, in the form of an 8-tuple as follows:

- the Twitter user ID
- the Twitter message ID
- the geo-coordinates of the message (either provided with the message, or automatically predicted)
- the probability of the predicated geolocation
- the text of the tweet

In addition to specifying a set of keywords for a given session, system users can dynamically select regions on the map, either via the manual specification of a bounding box, or zooming the map in and out. They can additionally change the time scale to display messages over, specify the refresh interval and also adjust the threshold on the geolocation predictions, to not render any messages which have a predictive probability below the threshold. The size of each place marker on the map is rendered proportional to the number of messages at that location, and a square is superimposed over the box to represent the maximum predictive probability for a single message at that location (to provide user feedback on both the volume of predictions and the relative confidence of the system at a given location).

References

Zhiyuan Cheng, James Caverlee, and Kyumin Lee. 2010. You are where you tweet: a content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 759–768, Toronto, ON, Canada. ACM.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL HLT 2011)*, pages 368–378, Portland, USA.

Marco Lui and Timothy Baldwin. 2011. Cross-domain feature selection for language identification. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011)*, pages 553–561, Chiang Mai, Thailand.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.