High performance mining of social media data

Judith Gelernter
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh PA 15213
gelern@cs.cmu.edu

Gang Wu
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
winston.gwu@gmail.com

ABSTRACT

News and disaster-related applications may benefit from real-time processing of large-volume, up-to-the-minute social media data. Our geo-mining algorithm finds local place references (of street, building, toponym and place abbreviation) in Twitter messages so that those messages can be put on a map. The ability to map is significant because it can present a timely overview of a situation. Our current research demonstrates that our prototype desktop algorithm that geo-locates Twitter messages with an F statistic of .90 accuracy for location identification will be viable on a large scale and in real time, for actual applications. We present methods of managing external resources, threading the algorithm and balancing the data load, that allow us to scale up the application without significantly re-writing the code.

Categories and Subject Descriptors

C.3 [Special-purpose and Application-based Systems.] Real-time and embedded systems.

General Terms

Algorithms, Performance, Experimentation

Keywords

Real time, data stream, multicore, multiprocessor parallel computing, Twitter, geo-location, geo-parsing

1. INTRODUCTION

Social media data may supplement traditional journalistic sources of information about events and crises. Twitter, in particular, has been mined for information about earthquakes [1], and about other emergencies [2],[3],[4].

Our earlier research demonstrated a prototype application that associated street, building, toponym and location abbreviation within Twitter messages to a high degree of accuracy [5]. Our current research experiments with methods to scale up that application with the help of greater computing resources so that it might be deployed in real time with much larger data volume.

Other geo-location algorithms rely on superficial methods that filter the Twitter data stream directly. This requires less data processing and is easily to scale. For example, the Twitter map provided by Tweography filters GPS location, that provided by Trendsmap filters by user-supplied location, and that provided by Twitris filters by both GPS and user-supplied location.[1] Our geo-location algorithm, in contrast, geo-parses within a metadata field (the tweet text), by comparison, and so requires more processing and a more complicated scale-up strategy.

The advantages of our deep geo-parsing algorithm is that (1) it allows the information rather than the user to be mapped, (2) it allows for mapping messages that would not otherwise be geo-locatable, (3) it allows for mapping at the geographical precision of street, and even building on a street. Such precision may not be as common in day-to-day messages as in crisis-related messages, but in mapping particular events that precision is useful.

Our data was pre-selected for language and disaster-potential from the 10% of the full Twitter stream as archived for Carnegie Mellon research. Our data set was collected during the 3-day climax of the 2011 fire in the Texas state capital of Austin. See Table 1 for statistics.

Table 1: Geo-locatable messages: What can our geo-parsing algorithm add?

Austin fire tweets, N=3111	
Percentage of tweets that are geo-locatable by any means (User-supplied location, GPS, geo-words in tweet text)	63%
Percentage of geo-locatable tweets that have location in tweet text	32%
Percentage of geo-locatable tweets that can be located to city scale or more precisely via tweet text	13%
Percentage of tweets that include GPS location	1%

XSEDE'12, July 16-20, 2012, Chicago, Illinois, U.S.A.

[1] Tweography.com; Trendsmap.com; Twitris.knoesis.org/election

2. ARCHITECTURE

Presently, the system takes tweets selected by keyword and date (from the Austin fire), and outputs only those tweets that have associated locations. This is because it has been found that tweets that do not have locations are less informative [6]. Geo-coding by gazetteer look-up, supplemented by API calls to GoogleMaps, will allow those tweets to be placed on a map.

3. METHOD to SCALE UP

Our work follows the applicable requirements of real-time stream processing set out by [7]. Our goal was soft real-time processing. Our method was to (1) re-structure the algorithm and re-organize external resources, (2) thread the algorithm among the 16 cores of our supercomputer allotment, and (3) balance the data load among threads to the cores.

3.1 Re-structure the algorithm and resources

We have re-structured our desktop geo-parse algorithm by combining the formerly separate abbreviation identification module with the street, building and toponyn module.

Our external resources consisted initially of API calls to the Named Entity Recognizer OpenCalais, and the use of a third party spell check algorithm. We substituted the Stanford NER program which we could download into the resource library, and removed the spell-checker altogether, which caused us to lose some accuracy but gain immensely in speed. We also re-organized the gazetteer into an inverted list to improve efficiency.

Initializing external resources is time-consuming. However, that relative initialization time decreases proportionally with the volume of data to be processed. In our Austin tweet set, we found that 55% of the run time was dedicated to actual geo-parsing, but when we processed ten times the tweet set (about 30,000 tweets), the relative time for initialization diminished and 91% of the time was dedicated to geo-parsing.

3.2 Threading

Threading creates sub-processes that run in parallel. The number of threads possible is application specific. The upper limit for thread efficiency is a function of the way the threads cross-communicate, and the generation of each thread, as well as allocating resources (shared among threads), and scheduling. We time-tested processing from 1 to 12 threads. Run time per thread, that is, the time it takes for the data to be processed and for the thread to terminate, is the lowest when we have 10 threads.

3.3 Load balancing

We defined two load strategies based on the length of a tweet file. We tested processing all the short tweet first (which we call "EasyFirst"), and balancing short with long tweets (which we call "Balanced") with three trials per option (Table 2). Tests showed that the Balanced processed at about the same speed as the EasyFirst, even up to 16 threads (Table 2 shows only up to 5 threads). The advantage of the EasyFirst is that some results are returned as soon as they are processed, which improves response time for users.

Table 2: Three runs for each load strategy, showing how number of threads influences processing time.

Load strategy	1 Thread—time in millisec	3 Threads—time in millisec	5 Threads—time in millisec
EasyFirst	3182	7812	Average time 25662
EasyFirst	5039	11120	
EasyFirst	3159	7639	
Balanced	7561	8334	Average time 26563
Balanced	16031	16428	
Balanced	16593	17337	

4. DISCUSSION

Methods presented here allow us to process about 375 tweets per second. The Twitterverse as of March 2012 produced about 4000 tweets per second, according to TNW Social media. Of these, only about 2000 are in English, and tweets input into our geo-parse algorithm would be only those that contain specified keyword(s), leaving only a minor subset of that 2000. We therefore are approaching the speed that our algorithm would require to geo-locate tweets in real time.

Further experimentation directions might include altering the data structure of external resources in addition to the gazetteer, pre-processing of tweets to tag individual words such as stop words and prepositions for improved accuracy, as well as implementing more syntactic methods for geo-parsing that would be likely to hold across languages. Continuing testing of algorithm accuracy on large scale data sets will use for validation GPS location and user-supplied location metadata.

5. CONCLUSION

We have experimented on Blacklight, an Altix UV supercomputer, to scale up a desktop geo-parsing algorithm for faster service in real time. Algorithm restructuring and resource re-organization, threading and load balancing contribute to improving output performance.

6. ACKNOWLEDGMENTS

Our thanks to our Department Head, Professor Jaime Carbonell, as well as Nick Nystrom, Sergiu Sanielevici and Joel Welling from the Pittsburgh Supercomputing Center who have supported this research. Our development team includes Shuguang Han and Wangzi He, with past members Shilpa Balaji, and Nikolai Mushegian. Our data set obtains from the generosity of Carnegie Mellon Twitter data curator, Brendan O'Connor. Supercomputer time was allocated through a PSC grant, and continued research is funded by DARPA.

7. REFERENCES

[1] Sakaki, T., Okazaki, M., Matsuo, Y. (2010). Earthquake shakes Twitter users: Real-time event detection by social sensors, *Proceedings of the 19th international conference on world wide web*, 851-860.

[2] Yin, J., Lampert, A., Cameron, M., Robinson, B., and Power, R. (2012). Using social media to enhance emergency situation awareness. *IEEE Intelligent Systems*, Retrieved June 14, 2012 from http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=06148196

[3] Sreenivasan, N. D., Lee, C.S. and Goh, D. H-L (2011). Tweet me home: Exploring information use on Twitter in Crisis Situations. A.A. Ozok and P. Zaphiris (Eds): *Online Communities, HCII 2011, LNCS 6778*, 120-129.

[4] Maxwell, D., Raue, S., Azzopardi, L, Johnson, C. and Oates, S. (2012) Crisees: Real-time monitoring of social media streams to support crisis management. R. Baeza-Yates et al (Eds.) *ECIR 2012, LNCS 7224*, 573-575.

[5] Gelernter and Balaji, (2012). An algorithm for local geoparsing of microtext. Under review for *GeoInformatica*.

[6] Munro, Robert. 2011 Subword and spatiotemportal models for identifying actionable information in Haitian Kreyol. *Fifteenth Conference on Computational Natural Language Learning (CoNLL 2011)*, Portland.

[7] Stonebraker, M., Cetinemel, U., Zdonik, S. (2005). The 8 requirements of real-time stream processing. *SIGMOD Record 34*(4), 42-47.