Trendminer: An Architecture for Real Time Analysis of Social Media Text

Daniel Preoţiuc-Pietro
Department of Computer Science
University of Sheffield, S1 4DP, UK
daniel@dcs.shef.ac.uk

Sina Samangooei
School of Electronics and Computer Science
University of Southampton, SO17 1BJ, UK
ss@ecs.soton.ac.uk

Trevor Cohn
Department of Computer Science
University of Sheffield, S1 4DP, UK
t.cohn@dcs.shef.ac.uk

Nicholas Gibbins and **Mahesan Niranjan**
School of Electronics and Computer Science
University of Southampton, SO17 1BJ, UK
{nmg,mn}@ecs.soton.ac.uk

Abstract

The emergence of online social networks (OSNs) and the accompanying availability of large amounts of data, pose a number of new natural language processing (NLP) and computational challenges. Data from OSNs is different to data from traditional sources (e.g. newswire). The texts are short, noisy and conversational. Another important issue is that data occurs in a real-time streams, needing immediate analysis that is grounded in time and context.

In this paper we describe a new open-source framework for efficient text processing of streaming OSN data (available at www.trendminer-project.eu). Whilst researchers have made progress in adapting or creating text analysis tools for OSN data, a system to unify these tasks has yet to be built. Our system is focused on a real world scenario where fast processing and accuracy is paramount. We use the MapReduce framework for distributed computing and present running times for our system in order to show that scaling to online scenarios is feasible. We describe the components of the system and evaluate their accuracy. Our system supports easy integration of future modules in order to extend its functionality.

1 Introduction

Online social networks (OSNs) have seen a rapid rise in number of users and activity in the past years. Microblogs or OSNs that focus on the sharing of short text messages are amongst the most interesting and challenging for linguistic analysis.

Example use cases of such analysis include: political parties who are interested in monitoring microblogging platforms. They may want to track in real-time elements like user sentiment towards their cause. They may also be interested in frequencies of certain terms (e.g. politician names), emerging topics and the volume of discussion on a subject. Also, they can use these or other features in order to predict poll or election outcomes or to estimate the impact that a certain policy would have on the public.

For the last few years there has been a great deal of research interest in OSN data. Previous applications include predicting real world events like political polls (O'Connor et al. 2010) or financial markets (Bollen, Mao, and Zeng 2011). We expect our system to become a valuable resource for corporate users and researchers alike.

There are several challenges for a text processing tools when faced with OSN data. These include the short length of messages, inconsistent capitalization patterns, ad-hoc abbreviations, uncommon grammar constructions and threaded discussions of friends in a network structure. When standard text processing tools, like part-of-speech taggers or named entity recognition systems, have been applied to OSN data their results have shown a significant drop in accuracy (Gimpel et al. 2011; Ritter et al. 2011), making their use in a pipeline untenable. Several researchers have addressed some of these problems in the past years, creating specific text processing tools for OSN data. However, all of these tools were developed separately and with no explicit emphasis on how they would adapt to real life scenarios that include processing batches of millions of items or online data processing.

We propose a framework that can combine existing tools whilst being extensible to allow the addition of future components. Moreover, the system is built in as pipeline with interchangeable modules which gives the end user control over what processing steps are required for their given application. In order to achieve this, we keep the format of the original data and at each step of the pipeline we augment the output of the previous steps with extra fields corresponding to the results of preceding steps. The system is built using Twitter data and the JSON data format, but can be easily adapted to data from other OSNs (e.g.Facebook/Google+ status updates) that share similar features.

Due to the challenges posed by massive datasets and by the I/O bound nature of the analysis our setting lends itself perfectly to using in the MapReduce distributed framework. Text analysis tasks can mostly be done in parallel in the *map* part while the aggregation into feature vectors, which will form an essential part of the system in the future, is achieved in the *reduce* part of the framework.

In section 2 we discuss the architecture of our system. We are developing the system for two different use cases: one focused on batch processing of collections of millions of tweets and the other focused on online processing. In section 3 we present our existing modules and suggest a list of the modules to be integrated in our future work. We evaluate the accuracy of our modules and their running times in section 4 and we present plans for future work in section 5.

2 Architecture

The main goal in this work is the creation of a pipeline of preprocessing tools which we have identified as being useful in many Twitter analysis activities. We identified two main use cases. Firstly, the batch analysis of several terabytes of tweets and applying filters for keywords, language, etc in order to compute aggregate counts of features, sentiment, etc over them when dealing with archival scenarios. Secondly, the analysis of millions of tweets per hour when dealing with real time analysis scenarios. To address these concerns, we propose a set of command line tools. The tools implement the stages of our preprocessing pipeline, the stages of which we explore in more detail in Section 3.

We expect that any particular task in the pipeline applied to an individual tweet will have little processing requirements as compared to the I/O requirements of reading, preprocessing and outputting in a useable format several terabytes of compressed tweets. This *I/O bound* nature of twitter analysis has been addressed in the past (both by various authors as well as Twitter's own in house development team[1]) with the use of clusters of machines with shared access to distributed tweets using the MapReduce framework and distributed filesystem. MapReduce is a software framework for distributed computation. MapReduce was introduced by Google in 2004 to support distributed processing of massive datasets on commodity server clusters. Logically, the MapReduce computational model consists of two steps, Map and Reduce, which are both defined in terms of $< key, value >$ pairs. The dataset being processed is also considered to consist of $< key, value >$ pairs. In the case of processing twitter data, our keys are null and our values are individual twitter statuses held in the standard Twitter JSON format, augmented with an extra analysis map. We discuss the data format consistency considerations in greater detail below.

We take advantage of the relatively mature Apache Hadoop[2] MapReduce framework to apply distributed processing to tweets. When interacting with Hadoop it is possible to dictate the map and reduce functions using either *Hadoop Streaming*[3] or writing custom Java tools interacting with the underlying Hadoop Java libraries. Hadoop streaming allows the specification of the mappers and reducers through POSIX like standard in and standard out enabled command line utilities. This allows for quick prototyping using any language the user wishes, but doesn't provide the flexibility exposed when using Hadoop as a library in Java. Instead we choose to implement a Hadoop enabled preprocessing tool written in Java. This tool exposes the various stages of our preprocessing pipeline as modes. The inner components of the tool are shared between two separate tools: a local command line utility (primarily for local testing purposes) and a Hadoop processing utility. The individual stages of the preprocessing pipeline are implemented in pure Java and exposed as modes in the tools. In the local utility, individual tweets are loaded one at a time and each selected pipeline stage is applied to each tweet as it is loaded. In the Hadoop implementation the map stage is used to load each tweet wherein each preprocessing step is applied to an individual tweet and emitted by the Mapper and the Null reducer is used as no further processing needs to occur after map. The key consideration in the design of these tools are:

Modularity Our tools are engineered for extensibility. Firstly, the Hadoop and Local tools are both driven through the same "mode" specifications and implementations. To implement a new mode which works in both tools, a simple Java interface is implemented which specifies a single function which accepts a twitter status and adds analysis to the status's analysis field. Furthermore, multiple implemented modes can be executed in a single invocation of the tool. Concretely, this results in multiple analysis being performed on a single tweet while it is in memory (in the Local tool) or multiple analysis being performed in the single map task (in the Hadoop tool).

Data Consistency Related to modularity is the notion of data consistency. Components of the pipeline may run in isolation, or as a chain of preprocessing tasks. To this end each component must be able to predict what data is available and be able to reuse or reproduce the output of preceding stages it relies upon. Concretely this means that the original twitter status data must remain unchanged and instead the Twitter status JSON map is augmented with an "analysis" entry which is itself a map that holds all the output of the pipeline's stages. Implemented components of the pipeline use this analysis map to retrieve the output of previous stages[4] and they also add their own analysis to this map.

Reusability To guarantee the repeatability of all our experiments, we will make each stage of our preprocessing pipeline available to the wider community in a form which can be easily used to reproduce our results or achieve novel results in different experiments. To these ends the current versions of the both tools are made available in the OpenIMAJ multimedia library[5]. In doing so we allow third parties to access future releases and module extensions, as well as complete source code access and the ability to preprocess tweets following our methodologies.

3 Modules

As of the writing of this paper we have implemented and tested 3 components of our preprocessing pipeline in our analysis tools. All three stages are implemented in pure Java. These are: Twitter specific **tokenization**, short text **language detection** and **stemming**. We present a brief description of all these modules in the next subsections while a list of planned extensions to be added in the future to our system is presented in section 3.4.

[1]`http://engineering.twitter.com/2010/04/hadoop-at-twitter.html`

[2]`http://hadoop.apache.org/`

[3]`http://hadoop.apache.org/common/docs/r0.15.2/streaming.html`

[4]A stage has a unique name which it uses to store data.

[5]`http://openimaj.org`

3.1 Tokenization

Tokenization is an essential part in any text analysis system and is normally amongst the first steps to be performed in a preprocessing pipeline. Its goal is to divide the input text into units called tokens, with each of these being a word, a punctuation mark or some other sequence of characters that holds a meaning of its own.

Tokenization of OSN data and in particular of Twitter data poses some extra challenges than for traditional sources (e.g. newswire). In particular, we must handle URLs, sequences of punctuation marks, emoticons, Twitter conventions (e.g. hashtags, @-replies, retweets), abbreviations and dates. Our system handles all of these challenges as we show in the qualitative evaluation. Note that the current version only works with latin scripted languages and their conventions (e.g. delimitation between words by punctuation or whitespace) and an extension to other languages (e.g. Asian languages) is planned for future versions.

3.2 Language Detection

There are many language detection systems readily available to be used for this task. The main challenge for these when faced with OSN data is the short number of tokens (10 tokens/tweet on average) and the noisy nature of the words (abbreviations, misspellings). Due to the length of the text, we can make the assumption that one tweet is written in only one language. Most language detection tools work by building n-gram language models for each language and then assigning the text to the most probable language from the trained model.

We choose to use the language detection method presented in (Lui and Baldwin 2011) which we have reimplemented in Java. We choose this method over others for the following reasons: it is reported as being the fastest, it is standalone and comes pre-trained on 97 languages, it works at a character level without using the script information (this way we need to feed only the text field) and it was used by other researchers with good results (Han and Baldwin 2011).

3.3 Stemming

We use the traditional Porter Stemmer, which is the standard stemmer used in NLP and Information Retrieval tasks. We use the Snowball stemmer backed by the Terrier Snowball stemmer implementation.

3.4 Planned modules

In addition to the modules in the previous subsections, we have identified other tools that would be useful for users in order to analyze OSN data and build further analysis or systems based on this. We present a list of modules to be added in the future to our system, as well as available tools that can be integrated in case they exist.

Location detection is the process of assigning a tweet to a specific location. While some tweets are annotated with geolocation information, for most this information lacks. The main approach is trying to assign each user a *home* location, each tweet being thus assigned to the location of its author. Location detection is not trivial because most users don't disclose this type of information and this has to be inferred. There are two major approaches: either content analysis with probabilistic language models (Cheng, Caverlee, and Lee 2010) or inference from social relations (Backstrom, Sun, and Marlow 2010).

POS Tagging is the task of labeling each word in a text with its appropriate part of speech (e.g. noun, verb). Research has showed that state-of-the-art algorithms for texts decrease significantly in accuracy when faced with OSN data. A tool created for this type of data was developed in (Gimpel et al. 2011) and is freely available.

Named entity recognition is the task that extracts and classifies some of the tokens into categories like names of persons, organizations, locations, etc. (Ritter et al. 2011) presents the results of standard NER systems on OSN data and builds an improved freely available tool.

Normalizer for out-of-vocabulary words. Because of the noisy nature of the words in tweets there are many out-of-vocabulary words (abbreviations, misspellings of existing words) that can be mapped to vocabulary words. (Han and Baldwin 2011) presents an attempt to solve this problem.

Spam detection is the process by which we eliminate repeated texts used for promotion or that have irelevant content. Including these in our analysis can skew the feature distributions and bias future analysis.

Retrieval of messages related to a query that don't necessarily include the original query keywords. This is necessary on OSN data because of their short length and grounding in context, where users don't repeat all the information about a topic when expressing a point of view.

User influence can be established by the number of users receive one's message or how well their messages are spread. Services like Klout offer influence ratings for each user and we can make use of the public Klout API [6] to include these scores.

Sentiment can be computed using simple words lists like MPQA (Wiebe, Wilson, and Cardie 2005).

Aggregators into feature vectors for sentiment, named entities, tokens, etc. for varying time intervals.

4 Evaluation

In this section we focus on testing the accuracy of our implemented components of the system. In order to produce useful results, our system needs to perform its tasks both quickly and with high accuracy.

4.1 Tokenization

For the tokenization module we will evaluate the performance qualitatively by presenting some tweets that pose tokenization problems specific to microblogging text. Some representative examples are presented in Table 1 and we can conclude that our tokenizer handles OSN text very well.

[6] `http://developer.klout.com/api_gallery`

Table 1: Example tweet tokenisations

Tweet A	"@janecds RT _badbristal np VYBZ KARTEL - TURN & WINE&lt; WE DANCEN TO THIS LOL? http://blity.ax.lt/63HPL"
Tokens A	[@janecds, RT, _badbristal, np, VYBZ, KARTEL, -, TURN, &, WINE, <, WE, DANCEN, TO, THIS, LOL, ?, http://blity.ax.lt/63HPL]
Tweet B	"RT @BThompsonWRITEZ: @libbyabrego honored?! Everybody knows the libster is nice with it...lol...(thankkkks a bunch;))"
Tokens B	[RT, @BThompsonWRITEZ, :, @libbyabrego, honored, ?!, Everybody, knows, the, libster, is, nice, with, it, ..., lol, ..., (, thankkkks, a, bunch, ;))]

Table 2: Number of tweets (in millions) analyzed and created in an hour. Analysis performed: tokenization and language detection

Time	Local	Hadoop	10% Twitter	Total Twitter
1 hour	0.51	7.6	1	10

4.2 Language Detection

Language detection of short and noisy text has been shown to be a challenging problem. (Baldwin and Lui 2010) report a decrease in performance from around 90-95% down to around 70% with state-of-the-art language detection algorithms when restricting the input text's length.

We test the method that we integrated to our pipeline on the same microblog dataset used by (Carter, Weerkamp, and Tsagkias 2012). They report an accuracy of 89.5% when classifying into 5 different languages. Our accuracy is 89.3% on 2000 tweets using a 97-way classifier. For our setting, in which we want to assign texts to many languages, we conclude that our language identification system performs well, but with room for future improvement. Improvements are suggested in (Carter, Weerkamp, and Tsagkias 2012) where they use microblog specific information to improve accuracy to up to 97-98% when discriminating between 5 languages.

4.3 Running time

The results in Table 2 show timings of both our Local and Hadoop tweet preprocessing tools. Both experiments were run on tweets generated in one day on October 10th, 2010. The local tool was run on a single core whilst the Hadoop tool was run on a Hadoop cluster of 6 machines, totalling 84 virtual cores across 42 physical cores. Our timings show that on our relatively small Hadoop cluster our tools can preprocess tweets in the order of those created in a single day on Twitter in total [7], making our tools appropriate for analysing the tweets we have access to in real time. More importantly for our purposes, we can easily analyse 10% of the tweets generated per hour in under 10 minutes. Furthermore, due to Hadoop's ability to scale with the addition of new machines, we believe that the addition of a few machines will allow our tools to scale easily as Twitter grows in popularity.

[7] `http://techcrunch.com/2011/10/17/twitter-is-at-250-million-tweets-per-day/`

5 Conclusions and Future Work

We have presented a novel open source framework for performing text analysis tasks on OSN data. Our framework presents two modes of processing, batch and online, and is designed for fast and accurate processing in a distributed environment.

The preprocessing tools are constructed in a pipeline fashion. We demonstrated a few tools adapted to the specifics of OSN data and evaluated them, showing that new modules can easily be added to the pipeline or activated and deactivated based on the users needs. We presented results that indicate that the system can be scaled for online processing of streaming data.

Future work will concentrate on adding new modules for processing data based on the suggestions from 3.4. For both modes of usage, a future version of our system will contain a graphical interface with which users can visualize the data and the outcomes of the analysis. This will make our system open to be used not only by researchers, but also for commercial or home users for online exploratory analysis.

Acknowledgement

This research was funded by the Trendminer project, EU FP7-ICT Programme, grant agreement no.287863.

References

Backstrom, L.; Sun, E.; and Marlow, C. 2010. Find me if you can: Improving Geographical Prediction with Social and Spatial Proximity. In *Proc. WWW '10*, 61–70.

Baldwin, T., and Lui, M. 2010. Language Identification: The Long and the Short of the Matter. In *Proc. NAACL HLT '10*, 229–237.

Bollen, J.; Mao, H.; and Zeng, X. 2011. Twitter Mood Predicts the Stock Market. *J. Comp Sci* 2(1):1–8.

Carter, S.; Weerkamp, W.; and Tsagkias, E. 2012. Microblog Language Identification: Overcoming the Limitations of Short, Unedited and Idiomatic Text. *J. LRE*.

Cheng, Z.; Caverlee, J.; and Lee, K. 2010. You are where you Tweet: a Content-Based Approach to Geo-Locating Twitter Users. In *Proc CIKM '10*, 759–768.

Gimpel, K.; Schneider, N.; O'Connor, B.; Das, D.; Mills, D.; Eisenstein, J.; Heilman, M.; Yogatama, D.; Flanigan, J.; and Smith, N. A. 2011. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. In *Proc. ACL '11*, 42–47.

Han, B., and Baldwin, T. 2011. Lexical Normalisation of Short Text Messages: makn sens a #twitter. In *Proc. NAACL/HLT '11*, 368–378.

Lui, M., and Baldwin, T. 2011. Cross-domain Feature Selection for Language Identification. In *Proc. IJCNLP '11*, 553–561.

O'Connor, B.; Balasubramanyan, R.; Routledge, B. R.; and Smith, N. A. 2010. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In *Proc. ICWSM '10*.

Ritter, A.; Clark, S.; Mausam; and Etzioni, O. 2011. Named Entity Recognition in Tweets: An Experimental Study. In *EMNLP '11*.

Wiebe, J.; Wilson, T.; and Cardie, C. 2005. Annotating Expressions of Opinions and Emotions in Language. In *J. LRE*, volume 1.