Automatic News Generation Based on Twitter

Ali Alavi[1], Rolf Jagerman[1], and Tsay Kai-En[1]

ETH Zürich, Zürich, Switzerland
`alavis@ethz.ch, {rolfj, tsayk}@student.ethz.ch`

Abstract. In this report, we describe our approach towards generating news topics based on Twitter data. We tackle this problem by running a stochastic gradient descent classifier on a large set of news articles collected from different news agencies, and then using this classifier to classify Twitter posts into three news categories: sports, politics, and technology. Then we tokenize these classified tweets in order to extract names and nouns used in each tweet. Finally we perform a time series analysis on these set of words and recognize top ten trending topics as news-worthy. We present the results in a web based interface. The results, especially in politics and sports category, bear resemblance with the trending topics as reported by news agencies.

1 Introduction

We were motivated by a simple question: is it possible to generate more accurate and less biased news using Twitter data in comparison to traditional news agencies? Would such a system have a potential of becoming an alternative to mainstream news sources? If so, such a system could be a more reliable source of unbiased news, which in turn will have an immense effect on public awareness, knowledge and discourse.

Although there are many tools and websites, such as Google News, which automatically aggregate and present news articles, their data sources are mainstream news agencies. We, on the other hand, want to use public posts as our data source, hence using collective knowledge of citizens as our news agency. Hence, the citizens will be the audience as well as content providers.

In this model, every Twitter user can play a small, yet collectively significant role in news gathering, and hence in generating valuable news articles, even without his or her knowledge. This is the main difference between our vision and that of citizen journalism, where citizens intend to produce a news headline, article or story. A concept we would like to call *crowd-reporting*.

There is not much traditional techniques that can be used for realizing such system, since the core concept of this system makes use of big data collected from Twitter, something that no traditional technique can replace. Although we can think of using one or more data sources other than Twitter to achieve similar results.

2 Contribution

Current approaches to generate news based on Twitter focus on personal data for the most part. A system called "the tweeted times" [1] is the closest thing we could find to our idea. It generates a news paper type overview of the user's personal Twitter account. This personalized approach however does not attempt to extract interesting news topics from the large Twitter data stream. In our approach we first classify all available tweets and then perform time-series analysis on the results. A few works [3,8] applied different machine learning methods (Naive Bayes Classifier, Support Vector Machine, etc) to classify tweets, but their data size was relatively small compared to ours. Other works such as Trendsmap [7], show you the latest trends from Twitter for anyplace in the world. However, they analyze Twitter hashtags instead of the content of tweets. In our approach we classify the full contents of tweets into three categories and then automatically generate key terms of news based on these labeled tweets. This is a novel approach on which we could not find any previous research.

3 Data model

The Twitter streaming API provides data in JSON format. It outputs a tweet as a JSON object on a single line. We copied this data storage format to our permanent data storage. That is: we have a JSON object representing a tweet on each separate line in our data files.

The Spark Map-Reduce framework naturally works with text data where each line represents a new element. By using functionality such as Spark's `textFile`, we can map each line of text, which represents a tweet, to a specific function. This makes using the data in a Map-Reduce context extremely easy.

3.1 Data storage

We stored around 600GB of the Twitter stream data on Amazon S3. We used the s3cmd tool, which is a command line tool and client for uploading, retrieving and managing data in Amazon S3 [5]. By integrating our data collection script with this tool, we can directly upload the Twitter streaming data into the S3 bucket. This benefits the whole system because we can load the large data set using the S3 protocol into our Spark instances.

3.2 Data processing platform

The data processing platform we use is Apache Spark running on top of Amazon Elastic MapReduce (EMR). Initially we ran Spark on Amazon EC2 directly without using EMR. It turned out that the default configuration did not work well, and the executed tasks had a failure rate of about 50%. For this reason, we decided to try the EMR platform. This platform has Spark pre-configured, which resulted in a 0% failure rate for our tasks.

4 System architecture

The system architecture is depicted in Figure 1. The main idea here is to train a classifier for several news topics using the news Twitter accounts as training data. Then we use this classifier to predict the massive stream of public tweets into these news topics. Any tweets that are not classified as 'sports', 'politics' or 'technology' are discarded. This filters out most of the non-relevant tweets. Finally the predicted tweets are used to compute term frequencies over time and detect interesting news trends.

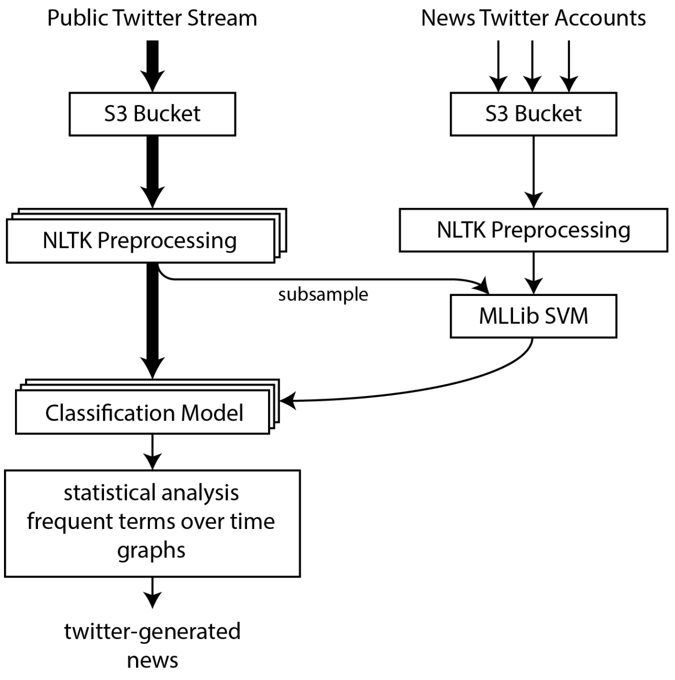

Fig. 1: The architecture of the system

The system is entirely built using Python [9] and we use the NLTK [4] and Scikit-Learn [6] libraries to do the preprocessing. For each tweet we extract the text, tokenize it, stem it and turn it into a sparse feature vector by using the hashing trick. The classifier is trained using Spark's MLLib [2]. Although we do not have sufficient training data to necessitate the use of MLLib's distributed learning, we use it to compute the predictions on our large Twitter data set, which requires scalability.

After classifying the tweets and discarding most of the non-relevant ones we are left with a smaller data set. A single computer is sufficiently capable of performing the time-series analysis on this data set in less than an hour. The final results of this analysis are then used by a simple web interface to display the Twitter generated news to the end user.

5 Performance Measurements and Results

It is important that classifying the tweets happens in a scalable and highly performant manner since we are classifying our entire data set (about 600GB). This classification stage discards most of the non-relevant tweets and the resulting data set is much smaller. A single computer is sufficient to run the time-series analysis on this smaller data set within an hour.

5.1 Execution time

The Spark MapReduce platform reduces our task into 10,702 smaller tasks. Each task gets approximately $\frac{600\text{GB}}{10702} = 57.4\text{MB}$ worth of data. Such a task takes on average 65.4 (± 20.6) seconds to complete, which means a single machine can process about 0.87MB of data per second. This includes parsing Twitter JSON, removing stop words and URLs, stemming, feature hashing and classification. On 20 machines it is possible to compute 17.54MB per second. In total this adds up to an execution time of 10 hours on the entire data set. Adding more machines would further decrease this.

5.2 Memory consumption

Our classifier uses feature hashing and stochastic gradient descent. Feature hashing ensures that each tweet gets hashed into approximately 1 million features. Because every tweet is processed individually by the stochastic gradient descent algorithm and our feature space is of a fixed size we achieve a space complexity of $O(1)$.

5.3 Solution quality

To measure the quality of our news classifier we do not train on a part of our dataset, and instead only test on that part. Furthermore we randomly select tweets from the public Twitter stream which we assume are not news-worthy or relevant. These tweets should not be classified in any of the categories. We are interested in optimizing the precision, recall and F_1-score. When the classifier scores high on these measures, we are confident in its ability to categorize new tweets.

The achieved F_1 score is on average 0.79. This is sufficient for the purpose of removing non relevant tweets. The time-series analysis performed on the resulting tweets is capable of handling a small amount of noise that the classifier might output.

class	precision	recall	f1-score	support
technology	0.78	0.92	0.84	25343
sports	0.75	0.73	0.74	25343
politics	0.83	0.75	0.79	25343
avg / total	0.79	0.80	0.79	76029

Table 1: Tweet classification performance over the three trained categories

6 Conclusion

The research question of this project is whether or not it is possible to predict the news using public tweets. In particular we are interested in the quality of the predicted news as it relates to existing news sources. We simplify this problem by focusing on three specific topics of news: "politics", "sports" and "technology".

The system we have constructed has provided us with several interesting insights. Political news gave us some of the best results. Obama's announcement about the new immigration law for example was very clearly reflected in the results (see Figure 2). Sports news seems particularly tailored to predicting when a match is ongoing and its score. News in the technology category gave the worst results, as it is for example very difficult to predict whether a tweet containing the word "iPhone" is actually relevant (such as the release of the new iPhone) or not interesting at all (people complaining about their iPhones).

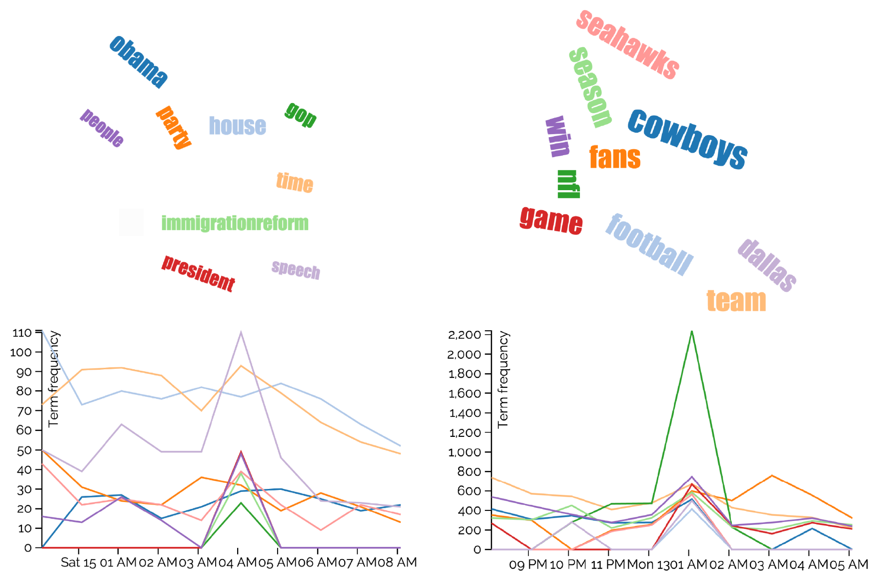

Fig. 2: Results on the categories politics and sports at two different times.

During the construction of the system we encountered several difficulties. After installing spark on amazon EC2, we noticed that when running a simple map-reduce program on a large data set about 50% of all tasks were failing and had to be restarted. As we were unsure what the cause of this problem was, we decided to do a clean install of Spark using Amazon's EMR. Because this Spark installation was specifically tailored for EMR, it worked much better and we had a 0% failure rate on the tasks. Another difficulty was collecting enough training data. Getting Twitter data is easy, but getting news-relevant Twitter data is rather difficult. We settled for using the news agencies' Twitter accounts as a source of training data. Overall we think more training data would have improved the solution quality. It was however difficult to obtain more training data as we exhausted all of the known large news agencies.

In retrospect we should have tried different and perhaps more specific categories for the classification of the news. In particular, the technology category made it very difficult to find any news-worthy topics.

Given more time it would be interesting to try additional features, such as the geolocation of tweets, the number of retweets, followers or other interesting measures. Also, the collection of more training data from sources other than Twitter could have improved the solution quality and is a worthwhile pursuit. Moreover, by collecting more Twitter data, we would be able to perform more accurate time-series analysis.

References

1. The tweeted times. http://tweetedtimes.com/, December 2014. [Online; accessed 15-December-2014].
2. The Apache Software Foundation. Mllib | apache spark. https://spark.apache.org/mllib/, Oct 2014. [Online; accessed 27-October-2014].
3. Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6, 2009.
4. Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.
5. Michal Ludvig. S3cmd tool for amazon simple storage service (s3). https://github.com/s3tools/s3cmd, Oct 2014. [Online; accessed 25-November-2014].
6. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
7. TrendMap. Trendmap. http://trendsmap.com/, Oct 2014. [Online; accessed 12-December-2014].
8. Jeff Tucker. twitter-classifier. https://github.com/trydionel/twitter-classifier, December 2014. [Online; accessed 12-December-2014].
9. G. van Rossum and F.L. Drake. Python reference manual, 2001. Available at http://www.python.org.