

Detecting Events in a Million New York Times Articles

Tristan Snowsill¹, Ilias Flaounas², Tijl De Bie¹, and Nello Cristianini^{1,2}

¹ Department of Engineering Mathematics, University of Bristol

² Department of Computer Science, University of Bristol

tristan.snowsill@bristol.ac.uk

Abstract. We present a demonstration of a newly developed text stream event detection method on over a million articles from the New York Times corpus. The event detection is designed to operate in a predominantly on-line fashion, reporting new events within a specified timeframe. The event detection is achieved by detecting significant changes in the statistical properties of the text where those properties are efficiently stored and updated in a suffix tree.

This particular demonstration shows how our method is effective at discovering both short- and long-term events (which are often denoted topics), and how it automatically copes with topic drift on a corpus of 1 035 263 articles.

Keywords: event detection, suffix tree, New York Times.

1 Introduction

In recent years we have witnessed an explosion of information through the ever-growing presence of the internet, the World Wide Web and more recently blogs and microblogs (*e.g.*, Twitter, Facebook statuses). There is a clear and present need for computational methods for processing the overwhelming amount of data which may be of interest to any of us.

Event detection and document summarisation methods can prove invaluable to people from all walks of life: journalists need to report the latest developments in current events, many of which are first reported online; investors need to very quickly assess the potential effect of breaking news on the values of their investments; companies need to monitor complaints and abuses of their products and services and many more uses.

One of the potential difficulties in identifying effective computational methods is in finding a dataset which is realistic in both content and scale. To the authors' knowledge the largest public test dataset for event detection is the TDT2 dataset with under 50 000 documents. To put this in perspective we have a system which indexes over 10 000 documents from the WWW every day.

Recently the New York Times released an annotated corpus of their articles from 1987 to 2007 with over 1.8 million documents [1]. This corpus is potentially an excellent testbed for computational methods as it is many orders of magnitude larger than existing corpuses and will truly test the ability of methods to scale.

2 Our Method

Our method is described in [2], so we summarise our method here. As articles from a text stream are observed they are added to a *generalised suffix tree*, a data structure which efficiently indexes the substrings of a set of texts. We annotate the suffix tree at its nodes with sufficient statistical information to perform an hypothesis test for every node (and due to the suffix tree structure for every n -gram in the text). Each hypothesis test gives the extent to which the observed frequency of an n -gram has increased from its expected frequency (calculated as a weighted average of its observed frequency in the past) in the form of a p -value. n -grams with a lower p -value have undergone a more significant increase in frequency. By ranking n -grams in order of ascending p -value we see the most significant n -grams which are indicators of events.

The suffix tree of a text of length n (or a set of texts with a combined length of n) can be stored in $\mathcal{O}(n)$ space. The suffix tree can be built in $\mathcal{O}(n \log |\Sigma|)$ time, where Σ is the *alphabet* (the set of symbols from which documents are drawn). In many applications of suffix trees $|\Sigma|$ is very small and $\mathcal{O}(n)$ construction time is often given, *e.g.*, for DNA, $\Sigma_{\text{DNA}} = \{a, c, g, t\}$. In our method $|\Sigma|$ is very large as it is the set of all words and punctuation symbols observed in the text as the text is tokenized at word boundaries and punctuation symbols. Our construction is a modification of Ukkonen's method [3] designed to be more efficient when the suffix tree is stored in external memory (*i.e.*, on a hard disk).

3 The Dataset

We considered articles from the New York Times annotated corpus from 1st January 1996 to 18th June 2007. We used only the *Body* attribute of each article (*i.e.*, we did not include the headlines or any tags). There are 1 035 263 articles in this time period with a total size of 3 401 MB and a length of 793 500 772 symbols after tokenization with an alphabet containing 1 422 974 symbols (stored in 12 MB).

4 Results

The suffix tree constructed from the corpus was 36 216 MB in size, a 10.6-fold increase in size over the original corpus text. Table 1 gives further details of the suffix tree.

We computed the p -value for each n -gram by comparing different weighted averages of the frequency of the n -gram. The weighted averages used were exponentially weighted averages,

$$\hat{x}_i = \alpha \hat{x}_{i-1} + (1 - \alpha) x_i,$$

where α is a weighting parameter. We used six different exponential weighting parameters: $\alpha = \{0, 0.215, 0.518, 0.720, 0.858, 0.950\}$; these correspond to 99%

Table 1. The suffix tree created from the New York Times corpus

Sequences	1 035 263
Suffixes	793 500 772
Alphabet size	1 422 974 (12.1 MB)
Nodes	
Internal	206 163 152
Leaf	777 468 706
Size on hard disk	36 216 MB

decay after 0 (*i.e.*, memoryless), 3, 7, 14, 30 and 90 timeframes respectively. For each value of α we stored a weighted average of the number of articles containing each n -gram, and the total number of articles in the timeframe. In addition we stored a timestamp for each n -gram recording the last timeframe it was observed in, allowing for just-in-time updates of the weighted averages. These weighted averages allowed for efficient computation of a p -value for each n -gram for each pairing of α values.

The statistics were stored in C++ floats for real values and in C++ ints for integer values. The total space requirement for each node in the suffix tree for the statistics was therefore

$$[4n_\alpha + 8] \text{ bytes} = 32 \text{ bytes.}$$

We chose not to compute p -values for leaf nodes as their frequencies are almost always 0 or 1. The total space requirement for the statistics is then 6.14 GB and the *total* space requirement of our method was 41.5 GB.

We considered three time periods, 1996–2000, 2000–2004 and 1996–2004. For each of these time periods and for each pairing of α values we calculated the forty most significant n -grams. We show a selected subset of these in Table 2.¹ We also show the timelines of these n -grams for the entire duration in Figure 1.

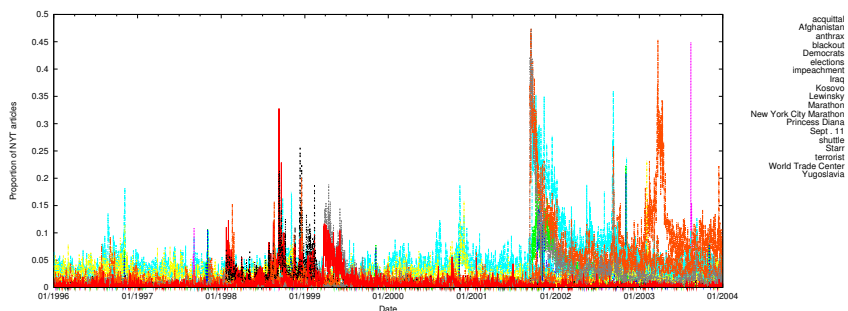


Fig. 1. Timelines of the most significant n -grams

¹ We believe it will be possible to automate this selection based on linguistic properties.

Table 2. The top five events detected in the three time periods using different comparison methods

Short term comparison method ($\alpha_1 = 0, \alpha_2 = 0.215$)		
1996–2000	2000–2004	1996–2004
New York City Marathon	Afghanistan	Afghanistan
Starr	Marathon	New York City Marathon
elections	terrorist	terrorist
Democrats	World Trade [Center]	World Trade [Center]
impeachment	Iraq	Iraq
Medium term comparison method ($\alpha_1 = 0.518, \alpha_2 = 0.720$)		
1996–2000	2000–2004	1996–2004
elections	World Trade [Center]	World Trade [Center]
Lewinsky	blackout	blackout
Iraq	elections	elections
acquittal	terrorist	terrorist
New York City Marathon	shuttle	Lewinsky
Long term comparison method ($\alpha_1 = 0.858, \alpha_2 = 0.950$)		
1996–2000	2000–2004	1996–2004
elections	World Trade Center	World Trade Center
impeachment	Sept. 11	Sept. 11
Kosovo	elections	elections
Yugoslavia	blackout	blackout
Princess [Diana]	anthrax	anthrax

5 Discussion

We have demonstrated our method for event detection on a large publicly available corpus of text. The method draws from areas of hypothesis testing and combinatorial pattern matching to solve a text stream mining problem.

Further results will be shown on the website for the project.²

References

1. Sandhaus, E.: The New York Times Annotated Corpus. In: Linguistic Data Consortium, Philadelphia (2008)
2. Snowsill, T., Nicart, F., Stefani, M., De Bie, T., Cristianini, N.: Finding surprising patterns in textual data streams. In: 2010 IAPR Workshop on Cognitive Information Processing (2010)
3. Ukkonen, E.: On-line construction of suffix trees. *Algorithmica* 14(3), 249–260 (1995)

² http://memewatch.enm.bris.ac.uk/new_york_times/