Signalling Events in Text Streams

Jelle Schühmacher and Cornelis H.A. Koster

Institute for Computing and Information Sciences (ICIS)
Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
{j.schuhmacher,kees}@cs.ru.nl

Summary. With the rise of Web 2.0 vast amounts of textual data are generated every day. Micro-blogging streams and forum posts are ideally suited for signalling suspicious events. We are investigating the use of classification techniques for recognition of suspicious passages.
We present *CBSSearch*, an experimental environment for text-stream analysis. Our aim is to develop an end-to-end solution for creating models of events and their application within forensic analysis of text-streams.

Key words: Signalling Events, Interactive learning, Text Mining, Text Streams

1 Introduction

We are investigating the use of classification techniques for signalling potentially suspicious events in passages coming from a stream of text. Micro-blogs and forum posts are ideally suited for signalling possibly suspicious events. It is difficult to get a grip on this kind of content, because texts are typically short. Furthermore, they need to be processed fast, since signalling an event after the fact is not very useful, and may even prove catastrophic.

Depending on the severity of the event to be signalled, a balance needs to be found between the economic feasibility of manually processing false positives and missing out on suspicious events. Therefore, it is important that a forensic text-stream mining application provides support for deciding on such a balance in an informed and verifiable way.

Due to the dynamic nature of user generated content, new types of suspicious events may frequently emerge. Therefore, it is not enough to be able to signal known events in text streams in an automated way. A truly useful forensic software support system will also provide methods to quickly and interactively build new models of emerging events.

2 CBSSearch

We are currently in the process of developing *CBSSearch* (Classifier Based Stream Search), an integrated environment for text-stream analysis. Our aim is to create an end-to-end solution for automated text-stream analysis.

Existing systems differ from CBSSearch in that they tend to use keyword-picking or hand crafted rules to find events in text-streams, opposed to using classification techniques to build a model of a particular event. An event model in CBSSearch is a collection of terms with statistical weights. Later this model will be extended to use dependency triples, recent experiments show this could prove to be helpful [1].

With CBSSearch it is possible to interactively train a event model of a particular event of interest. The resulting model is used to continuously search in streams of text for passages that match the event model.

2.1 System Overview

CBSSearch comprises four major components, a stream indexer, a stream analysis component, an interactive learning component, and a support system for sifting through suspicious events.

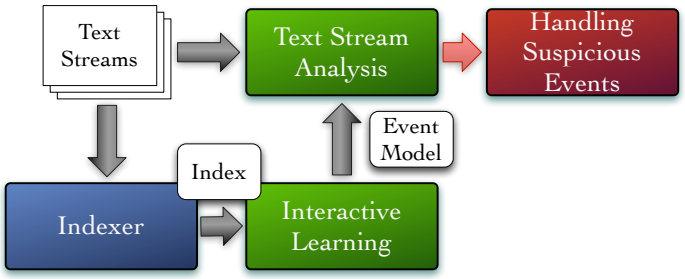

Fig. 1. Overview of CBSSearch

2.2 Stream Indexer

The *index* is used for interactively bootstrapping new event models from past examples. The Stream Indexer uses the Apache Lucene framework to efficiently create a database of examples. Apache Lucene is high-performance, scalable, full-featured, open-source, and written in Java. More information can be found on the official Lucene website[1]. In our testing Apache Lucene proves to be fast enough for our purposes.

2.3 Interactive Learning

The purpose of the *Interactive Learning* component is to train new event models with a minimum of effort. The component provides means for creating an event model by semi-supervised bootstrapping [3] from examples captured by the indexer.

[1] `http://lucene.apache.org/java/`

Training Bootstrap A training bootstrap starts with a seed, a set of positive and negative examples provided by a human operator. An initial event model is trained using the *Linguistic Classification System*[2]. The initial model is then iteratively refined by letting the system choose the best next training document which is presented to the user for judgement.

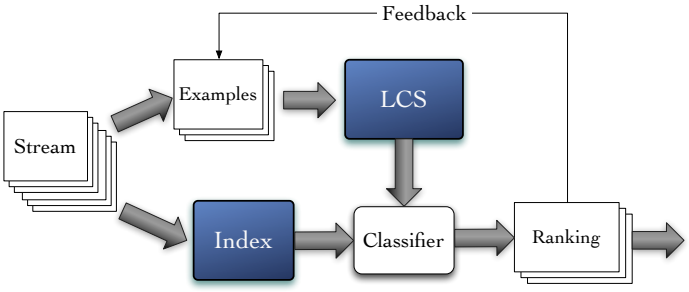

Fig. 2. The bootstrap process

The LCS is a basic component for all applications involving document classification, developed in the course of the ESPRIT projects DORO and PEKING, with a proven track record in the field of patent classification [2].

Active Learning The number of manual judgements to be made is kept low by using active learning [4]. In CBSSearch active learning means the system makes a classification using the event model under construction, from this classification a the set of passages is selected for which the classifier is least certain they fit in this event model. These passages are presented to the operator for judgement using the same method as the stream analysis component. Passages that score above a certain threshold are assumed to be judged correctly by the classifier and are automatically added as training examples.

2.4 Stream Analysis

The resulting event models, in the form of linear classifiers, are used used to perform *on-the-fly classification* of stream passages. When a segment of text in the stream contains a combination of terms present in an event model for which the weights sum up to more than a threshold defined by the operator, the segment is signalled as suspect. The operator has control over precision and recall by adjusting the threshold.

The system reports suspicious passages to a human operator, who will then be able to see why a passage has been signalled as suspect by means of a document popup. The popup displays the segment with terms that occur in the event model highlighted.

[2] `http://www.phasar.cs.ru.nl/LCS/`

3 Preliminary Results

The CBSSearch system is still under development. We have a made preliminary analysis of the performance of stream classification using an artificial example derived from pre-classified documents. In order to investigate the best strategies for stream classification we measured whether event model complexity and the number of training examples could be kept low. The number of training examples which have to be judged manually must be kept as low as possible in order to reduce the amount of manual labour needed.

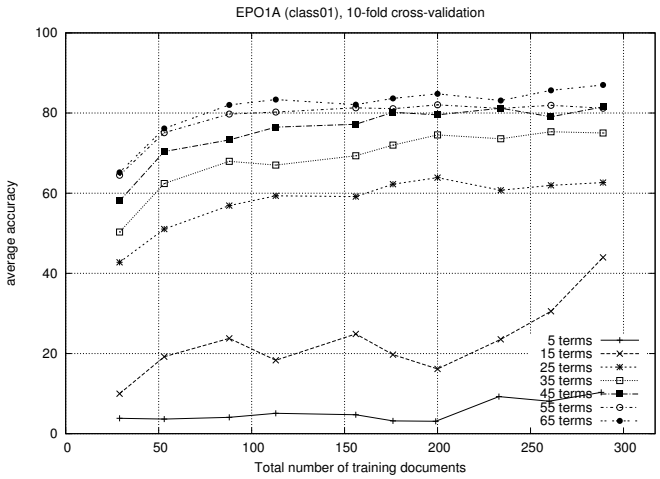

Fig. 3. Learning curves with increasing number of terms

This graph shows the trade-off between model complexity and the number of train documents. Fir this example a reasonable number of training examples (less than 100) suffices to achieve an acceptable accuracy (F1-value) as a starting point. Further training can then be performed using feed-back.

We are currently implementing CBSSearch and are collaborating with police authorities to find a suitable application for which training material can be easily obtained.

References

1. Jean G. Beney Cornelis H.A. Koster. Phrase-based document categorisation revisited. In *CIKM'09: Proceedings of the 18th ACM Conference on Information and Knowledge Management (to appear)*, 2009.
2. Cornelis H. A Koster, Marc Seutter, and Jean Beney. Multi-classification of patent applications with winnow. *Lecture Notes In Computer Science*, 2890:111–125, 2003.
3. Cornelis H.A. Koster, Paul Jones, Merijn Vogel, and N.Gietema. The bootstrapping problem. *2nd Workshop on Operational Text Classification Systems*, 2002.

4. David Lewis and William Gale. A sequential algorithm for training text classifiers. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, 1994.