International Journal of Scientific Engineering and Technology
Volume No.4 Issue No.10, pp: 478-480

ISSN :2277-1581
01 Oct. 2015

Text Stream Mining Using Suffix Trees

Pamela Vinitha Eric, Kusum Rajput, Veda .N
RGIT, Bangalore.
Email ID: pamela.vinitha@gmail.com

Abstract: Information explosion has resulted in the need for more advanced methods for managing the information. Text stream mining, is very important as people and organizations are trying to process and understand as much of information as possible. Generalised suffix tree is a data structure which is capable of solving a number of text stream mining tasks like detecting changes in the text stream, identifying reuse of text and detecting events by identifying when the frequencies of phrases change in a statistically significant way. An efficient method with polynomial time complexity that uses suffix trees to analyse streams of data in an online setting is discussed.

Keywords: Data stream, tree, mining, n-grams.

I. Introduction

Massive text streams are created by interaction of individuals over social networks or by structured creation of particular kinds of content by dedicated organizations like news-wire services[1]. The availability of such vast amount of information makes it necessary to have efficient methods to prioritise and filter these information as people are faced with the problem of missing out potentially important facts. Data mining techniques can be used to extract interesting and non-redundant information. Text mining can be be used to automatically generate content in the form of document summarisation and more advanced techniques[13]. Data streams are sources which produce high volume of continuous data that arrive at a rapid rate and its distribution changes on the fly. The application of data mining and machine learning tasks on data in the form of data streams is known as data stream mining. Text stream mining provides a set of methodologies and tools for discovering, presenting and evaluating knowledge from streams of textual documents.

II. Related Work

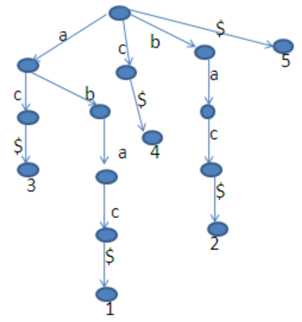
Kleinberg [8] used techniques from Hidden Markov Models (HMMs) to model the behaviour of a text stream whereas Leskovec et al[9] identified quoted phrases in articles and tracked them throughout a selected corpus, by clustering them together based on their edit distance after filtering out any which were not frequent enough. Whitney et al.[16] used statistical hypothesis tests to create an on-line event detection algorithm. Keogh et al[7] employs suffix trees to find patterns in a database. Here the data streams are first encoded into an artificial text stream and then analysed using an Mth-order Markov model. Qiankun Zhao[19] detect events from social text streams by exploring the content as well as the temporal, and social dimensions. Text streams are represented as multigraphs, each node represents a social actor and each edge represents a piece of text communication that connects two actors. Events are detected by combining text-based clustering, temporal segmentation, and information flow-based graph cuts of the dual graph of the social networks. Xuanhui Wang et al[14] proposed a general probabilistic algorithm to discover correlated bursty patterns and their bursty periods across text streams even if the streams have completely different vocabularies.

III. Suffix trees

Suffix tree is a powerful data structure first described by Weiner et al[15]. It is a compact representation of all suffixes of a given string. It is a rooted, directed tree that has n leaves labeled from 1 to n, and its edges are labeled by characters of the alphabet Σ[18]. A suffix trie is a trie storing all possible suffixes of a string S followed by \$ where \$ € ∑. The termination character \$ is inserted at the end of the string so that no suffix of *S* matches a prefix of another suffix of *S*. Every path from the root to a leaf represents a suffix of S. The key feature of the suffix trie is that for any leaf *i*, the concatenation of the edge labels on the path from the root to the leaf *i* exactly spells out the suffix of S that starts at position *i*. We denote S[i . . . n] by S_i. Every suffix of S is represented by some path from the root to a leaf. The labels of all edges outgoing from a node begin with different characters. Each node in the suffix trie is either the root, a leaf, or a branching node[10].

The leaves of the suffix trie also indicate the starting position of the suffix that ends at that leaf. Fig 1 shows the suffixes and the suffix trie of the string S= abac\$.

A Suffix tree for a given text is a compressed suffix trie for all suffixes of the given text. A compressed suffix trie T for a set



Index	Suffix
1	abac\$
2	bac\$
3	ac\$
4	c\$
5	\$

Fig 1: The suffix trie and indexes for the string S=abac\$

of strings S={S_1,S_2,…S_k} over alphabet Σ is a compressed representation of a trie where each internal node is forced to have at least two children. This can be achieved by combining edges which form a linear chain into a single edge with a string label. The Suffix tree for the above trie is shown in Fig2. Each internal

ISSN :2277-1581
01 Oct. 2015

node could have variable number of children, but the number of children is bounded by $|\Sigma|+1$.

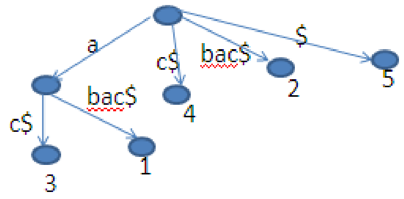


Fig 2: The suffix tree for S = abac$

Generalized Suffix Tree

A generalized suffix tree[6] indexes the suffixes of a set of documents, D= $\{S_1,S_2,\ldots\ldots, S_m\}$. Here $S_1,\ldots, S_m$ are m strings with lengths $n_1,\ldots, n_m$, respectively. Then the number of leaves of this tree will be

$$\sum_{j=1}^{m} n_j$$

Each leaf of a generalized suffix tree is labeled by the number j of the string and a number between 1 and n_j. The concatenation of labels on the path from the root to the leaf (i,j) represents the suffix $Sj[i, n_j]$ of the string j. Fig 3 shows the generalised suffix tree for the strings BANANA$ and BANDANA$.

Ukkonen's algorithm[17] is used to construct a suffix tree in linear time. The same can be applied to create a generalised suffix tree as well. Initially create a suffix tree for the first document, S_1 and later add the remaining documents one by one. Space complexity of such a tree is high and it can scale beyond the size of physical memory. Therefore a generalized suffix trees should be hard disk based[12].

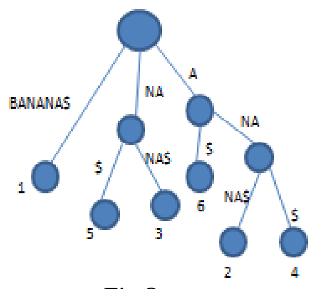


Fig 3: a

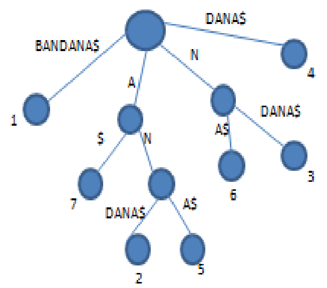


Fig 3:b

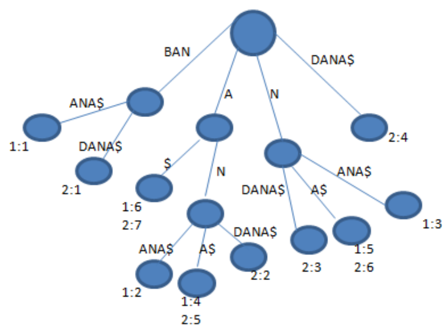


Fig 3: c

Fig 3: a The suffix tree for the string BANANA$
Fig 3: b The suffix tree for the string BANDANA$
Fig 3: c The generalized suffix tree for the strings BANANA$ and BANDANA$

Since the alphabet size is large linked lists are used to store outgoing edges. Each node has a link to its first child and to its next sibling or its parent if it is the last of its siblings.

IV. Suffix trees for stream mining

When text stream is processed in an on-line fashion space usage should be allowed to grow with the rate of the stream until it reaches some upper bound. This upper bound causes removal of some information to make space for the new. There are two ways to remove a document from a generalised suffix tree (GST). In the first method, given a GST 'T' and a document 'S', find the GST '(T – {S})'. This GST can be used remove old documents that are no longer likely to be used and the space can be reclaimed. The problem with this approach is that node IDs / substring relationship will be preserved. The second approach uses tandem suffix trees[13]. These are overlapping suffix trees which are constructed in parallel but the initial construction is staggered. When the suffix trees converge, the larger of the two is deleted and construction of a new suffix tree takes place. There is a clear upper bound on the space usage when the rate of text stream is constant.

Event Mining in Text Stream

Text mining focuses on mining a single text stream. This produces mixed mining results as it does not distinguish a global event from a local event. Hence text mining research should focus on multiple streams to detect latent events. Xuanhui Wang et al [14] has shown that mining correlated bursty topic patterns from coordinated text streams can help discover interesting common events that have influenced all the streams and can reveal interesting associations and linkages between the involved streams. It can also help discover local patterns more accurately by factoring out the global noise.

In text stream mining event detection deals with identification of the onset of external events which affect the behavior of the text stream[19]. Event detection is used when mining emails and to detect emerging topics in scientific literature. The most frequent application of event detection is in the study of news streams to identify news events[8]. James Allan et al [3]deals with on-line and retrospective event detection. On-line new event detection identifies new events in a stream of stories while retrospective detection identifies all of the events in a corpus of stories. In retrospective event detection it is assumed that each story discusses at most one event and can be included in at most one cluster only.

Suffix Trees and Event Detection

The set of all n-grams in the text stream can be indexed using a suffix tree. An n-gram is a string of n symbols drawn from the alphabet denoted as $\sum$. As all n-grams are indexed the behavior of the text stream can be tracked and appearance of any n-gram in large amount can be easily detected. Here the frequency of every n-gram in the text stream is tracked and different moving averages are used to understand the behaviour of an n-gram in the short and long term. The two moving averages are compared to

identify if an n-gram has become more frequent recently and assess the significance of this frequency. At the end of each timeframe, a batch of documents from the text stream are processed, tokenized and inserted into the generalised suffix tree[13].
A data stream of document frequency is created for each n-gram in the text stream. Document frequency gives the number of documents containing the n-gram at least once. At the end of each timeframe, p-value which specifies the significance measure is computed for each n-gram. The n-grams with the lowest p-values are the most significant and are chosen as the indicative events.
Each node of the GST stores an exponentially weighted moving average (EWMA) of that n-gram. Adding documents to the GST results in the creation of new nodes, some existing edges may be split as well. When an edge is split the new node on the edge inherits the EWMA of the node at the end of the edge. The nodes of the GST are numbered such that each node has an ID and proper mapping between each node ID and the corresponding string should be maintained. Similar n-grams are clustered together and is called a Memecluster[13]. In a MemeCluster constituent n-grams are mutations of a single n-gram, which is considered as a meme. Clustering is based on the positions of significant n-grams within the suffix tree. The distance between two significant n-grams, u and v, is defined as

$$D(u, v) = \begin{cases} 1 & \text{if u is a child of v or vice versa,} \\ 1 & \text{if u has a suffix link to v or vice versa,} \\ 1 & \text{otherwise.} \end{cases}$$

The n-grams are clustered together such that the distance between any pair of clusters is infinite. Each cluster has a representative n-gram which is selected by filtering out n-grams based on conditions like, brackets and quotes are balanced, n-grams do not start with inappropriate punctuation etc., and then selecting the longest remaining n-gram. In case more than one n-gram qualify to be the representative n-gram then the one with the lowest p-value is selected. MemeClusters do not describe an event correctly even though an event carries a number of memes, and these memes travel together. Hence the MemeClusters are grouped into EventClusters. Each EventCluster contains a number of memes which collectively define an event[4].
The set of articles which contain an n-gram x during the timeframe i is denoted as Ai(x). The Jaccard distance between the article sets of each pair of MemeClusters, (X1, X2), are calculated as

$$J_i(X_1, X_2) = 1 - \frac{|A_i(x_1) \cap A_i(x_2)|}{|A_i(x_1) \cup A_i(x_2)|}$$

Now hierarchical clustering methods are used to cluster MemeClusters together into EventClusters. Two EventClusters can merge if there is a single pair of MemeClusters with Jaccard distance less than or equal to 0.75.

V. Conclusion

Suffix trees enable us to track the frequency of every n-gram observed in the text stream through the use of EWMA thereby detecting events in the text stream. By calculating p-values for each n-gram it is possible to identify which n-grams have the most statistically surprising increase in frequency. This event detection method quickly identifies when customers have the same complaint and streamlines complaints handling.

References:

i. *Aggarwal, Charu C. "Mining text streams." Mining Text Data. Springer US, 2012. 297-321.*

ii. *Aggarwal, Charu C., and ChengXiang Zhai. Mining text data. Springer, 2012.*

iii. *Allan, James, et al. "Topic detection and tracking pilot study final report." (1998)*

iv. *Chezhian, V. Umadevi, Thanappan Subash, and M. Ragavan Samy. "Hierarchical Sequence Clustering Algorithm for Data Mining." Proceedings of the World Congress on Engineering. Vol. 3. 2011.*

v. *Chim, Hung, and Xiaotie Deng. "A new suffix tree similarity measure for document clustering." Proceedings of the 16th international conference on World Wide Web. ACM, 2007.*

vi. *Iliopoulos, Costas S., et al. "The weighted suffix tree: An efficient data structure for handling molecular weighted sequences and its applications." Fundamenta Informaticae 71.2 (2006): 259-277.*

vii. *Keogh, Eamonn, Stefano Lonardi, and Bill'Yuan-chi Chiu. "Finding surprising patterns in a time series database in linear time and space." Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2002.*

viii. *Kleinberg, Jon. "Bursty and hierarchical structure in streams." Data Mining and Knowledge Discovery 7.4 (2003): 373-397.*

ix. *Leskovec, Jure, Lars Backstrom, and Jon Kleinberg. "Meme-tracking and the dynamics of the news cycle." Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009.*

x. *Maaß, Moritz. "Suffix trees and their applications." Ausarbeitung von der. Ferienakademie 99 (1999).*

xi. *Rasheed, Faraz, Mohammed Alshalalfa, and Reda Alhajj. "Efficient periodicity mining in time series databases using suffix trees." Knowledge and Data Engineering, IEEE Transactions on 23.1 (2011): 79-94.*

xii. *Schürmann, Klaus-Bernd, and Jens Stoye. "Suffix tree construction and storage with limited main memory." (2003).*

xiii. *Snowsill, Tristan, et al. "Finding surprising patterns in textual data streams."Cognitive Information Processing (CIP), 2010 2nd International Workshop on. IEEE, 2010.*

xiv. *Wang, Xuanhui, et al. "Mining correlated bursty topic patterns from coordinated text streams." Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007.*

xv. *Weiner, Peter. "Linear pattern matching algorithms." Switching and Automata Theory, 1973. SWAT'08. IEEE Conference Record of 14th Annual Symposium on. IEEE, 1973.*

xvi. *Whitney, Paul, Dave Engel, and Nick Cramer. "Mining for Surprise Events Within Text Streams." SDM. 2009.*

xvii. *Ukkonen, Esko. "On-line construction of suffix trees." Algorithmica 14.3 (1995): 249-260.*

xviii. *Vásárhelyi, Bálint Márk. "Suffix Trees and Their Applications."*

xix. *Zhao, Qiankun, Prasenjit Mitra, and Bi Chen. "Temporal and information flow based event detection from social text streams." AAAI. Vol. 7. 2007.*