PROMINENT STREAKS DISCOVERY ON BLOG ARTICLES

by

JIJO JOHN PHILIP

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE & ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2012

To my mother Elizabeth Philip and my father Mohan Philip Thomas

who set the example and who made me who I am.

ACKNOWLEDGEMENTS

It is with great pleasure that I would like to thank those people without whom it would have been impossible to complete my thesis. First and foremost, I would like to thank my supervising professor Dr. Chengkai Li for constantly motivating and encouraging me, and also for his invaluable advice during the course of my research work.

I wish to thank my academic advisors Dr. Leonidas Fegarus and Dr. Sharma Chakravarthy for their interest in my research and for taking time to serve in my thesis defense committee. I am grateful to all the teachers who taught me during the years I spent in school, first in Kuwait, then in India and finally in the Unites States.

Finally, I would like to express my deep gratitude to my brother and sister-in-law who have encouraged and inspired me. I am extremely fortunate to be so blessed. I am also extremely grateful to my mother and father for their sacrifice, encouragement, patience through out my undergraduate and graduate studies. I would also like to thank several of my friends who have helped me throughout my career including my colleagues. I would like to thank my friend Nandish Jayaram for his valuable guidance and motivation.

November 14, 2012

ABSTRACT

PROMINENT STREAKS DISCOVERY ON BLOG ARTICLES

Jijo John Philip, M.S.

The University of Texas at Arlington, 2012

Supervising Professor: Chengkai Li

We are surrounded by data in various forms such as Twitter tweets, Facebook statuses, news, blogs and much more. Extracting meaning from such a massive collection of unstructured data would lead to interesting stories. Examples of such stories can be *who was the most popular actor in a particular month* or *which diseases were people most concerned about in year 2008*. In this thesis, we propose to discover popular entities mentioned in blog articles based on the concept of prominent streak. Given a sequence of values for a named entity (e.g., a person, a place, etc.) where each value is the occurrence frequency of the entity in blog articles during a corresponding period of time, a prominent streak is a long consecutive subsequence of only large(small) values. Whether a streak is prominent also depends on how it fares against streaks for comparable entities. Using the distributed data processing framework Hadoop, we find entity occurrences in a set of blog articles with a trie-based data structure. Prominent streak discovery algorithms are applied over the detected sequences of entities occurrences to derive interesting stories. Our experiments and evaluation are done over the ICWSM'09 Spinn3r blog dataset, which contains over 44 million blog articles for the months of August and September in 2008.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Motivation

In recent years, the prominence of social media in our day to day life has risen quite dramatically. The emergence of the World Wide Web (WWW) has caused massive amount of information being produced on a daily basis and to leverage that information for the users become quite tedious. Every person is a reporter, means anyone having any sort of a communication using the internet, could write an article, watch a game, post a video clip, comments on pictures etc at any moment of time. The emergence of social media on the WWW has grown exponentially in the last couple of years. Some of the examples are Facebook, Twitter, Flickr, Orkut and Blogs etc which takes the limelight making its user spend countless hours in a day on their web portals and share informations among them. These massive amounts of data have grown quite monstrously big and too large for humans to comprehend, and thus helped in the rise of the data mining tools for performing automatic data analysis and pattern discovery.

In contrast to the usual data mining, where one looks for patterns and knowledge in structured databases, text mining deals with unstructured, or semi-structured, textual data such as reports, e-mails, blogs, tweets or web-pages. This research will focus on a special case of text mining, namely entity discovery. If a sentence contains the occurrence of an entity, it has to be identified, this is known as entity discovery. This discovered entity is used to extrapolate patterns necessary to making meaningful information.

The need to find meaning in these massive blogs helps us to come up with interesting stories and focus on events which occur in our day-to-day lives. An interesting examples would be to find the most popular presidential candidate before the elections. Another example is to find the 10 most common diseases people are concerned and talked about during a particular interval of time. All these patterns make interesting stories to discuss.

1.2 Summary of the problem

The problem we focus in this research is to analyze a given unstructured dataset such as blog articles to find if a named entity (e.g., a person, a place, etc.) exists in it and whether any interesting stories can be found. Furthermore, compare stories for similar entities to find which one is more interesting than others and to identify the top-*k* prominent ones.

1.3 Challenges

Before going into more intricate details about the field of interest, the following section would give an overview of some of the challenges we faced dealing with the problem.

1. Can a large magnitude of unstructured data be used for our research and experiments?
2. How could we discover whether an entity is talked about in a blog article or not?
3. Is there any information that can be derived from a specific entity?
4. How do we find meaningful stories such as the most popular/talked about Actor or Politician during a particular time period from an unstructured data source?
5. Is it possible to determine whether an entity could be prominent over the other? If so how would we be able to differentiate between them?
6. How do we keep track of the list of entities in the order of their popularity?

1.4 Contribution

The contribution in this research is to analyze a given unstructured data such as blog articles in an efficient manner using a trie-based data structure for entity discovery. The trie-based data structure increases the efficiency for discovering an entity over the hashing-based technique. Using a distributed framework such as hadoop, finding the count of occurrences for any given entity over a specific period of time makes it simple and scalable. Furthermore, we use prominent streaks algorithm to find various interesting stories in the blog articles which is quite an effective technique to determine patterns and find out the most prominent entities among other comparable entities.

1.5 Organization of the thesis

Chapter 2 talks about the background and related works, Chapter 3 explains the formal problem definition and Chapter 4 describes about the approaches that has been taken to identify different prominent streaks. Furthermore, Chapter 5 discusses the experiments and evaluation.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Analytics of social media

Social media can be defined as “interactive platforms via which individuals and communities create and share user-generated content” [1]. Social media in our day-to-day life plays vital roles to socialize and mediate human interaction. Kietzmann studies the bursty and hierarchical structure in temporal text streams. The goal of this paper was to find how high frequency words change over a period of time. The word usage in multiple text streams, such as email, news articles and research publications, mostly displays some bursty and hierarchical behaviors. During certain time intervals, a few words appear more often than others and their frequencies change over time. Kleinberg assumptions consider the gaps between two consecutive messages follow an exponential distribution and uses infinite-state automaton to model the different levels of burstiness in different time scales. Words with significantly higher frequency than others are the words with high burstiness. Work done on similar lines but on multilingual streams of data is by Alexander Kotov [2]. While Wang et al. [3] and Kleinberg [1] focus on bursty behaviors and modeling, our focus is a high-performance algorithm to detect bursts across multiple entities. Our definition of a burst is simply a high value of frequency over a small interval of time, which is a straightforward definition with many applications in real world.

2.2 Distributed systems and frameworks

Distributed Computing in a science which deals with computation on multiple systems. A distributed system consists of multiple autonomous computers that communicate through a computer network. The computers interact with each other in order to achieve a common goal.

According to Wikipedia, MapReduce is a programming model for processing large data sets. MapReduce is a framework for processing embarrassingly parallel problems across huge datasets using a large number of computers (nodes), collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogenous hardware). Computational processing can occur on data stored either in a filesystem (unstructured) or in a database (structured). MapReduce can take advantage of locality of data, processing data on or near the storage assets to decrease transmission of data.

There are various type of distributed frameworks such as Hbase which is a Big table-like structured storage system for Hadoop HDFS, Apache Pig is another high-level data-flow language and execution framework for parallel computation and processing which is built on top of Hadoop Core. Hive a data warehouse infrastructure which allows SQL-like adhoc querying of data (in any format) stored in Hadoop, Mahout which is a scalable Machine Learning algorithms using Hadoop, Cassandra is a scalable multi-master database with no single points of failure.

Apache Hadoop is one of the most popular framework for running applications on large cluster built of commodity hardware. The Hadoop framework transparently provides applications both reliability and data motion. Hadoop implements a computational paradigm named MapReduce, where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster. In addition, it provides a distributed file system (HDFS) that stores data on the compute nodes, providing very high aggregate bandwidth across the cluster. Both MapReduce and the

Hadoop Distributed File System are designed so that node failures are automatically handled by the framework.(from [4])

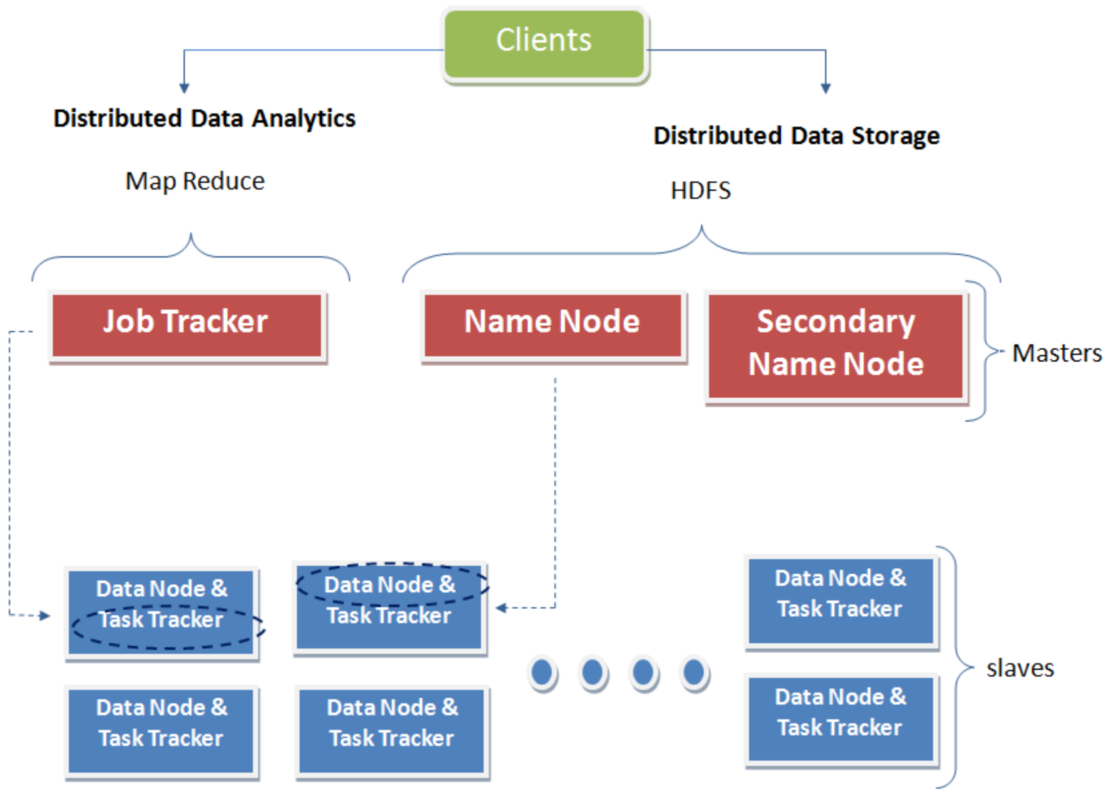

Figure 2.1. MapReduce Workflow

Hadoop framework consists of two major components which are distributive data analytic and distributed data storage. The distributive data analytic part is where the job tracker maintains and keeps track of the task trackers which are present in the slave nodes. The distributed data storage part is the hadoop distributed file system (HDFS) where the meta data and other details about the data, the slave configurations, which slaves are working and the details about where the partitioned data is preserved. The main components in the distributive data storage is the namenode and the secondary namenode.

In the distributive data analytic part, the slave nodes uses the task tracker to execute the mapper and reducer programs. The mapper collects the input from the HDFS and runs the program given by the user, once the mapper part of the program is run the output is passed to the reducer. The phase where the output emitted by the mapper is collected and passed to the reducer as the input is known as the shuffle phase. The reducer then does the job/program given by the user to give the final output into the HDFS. In the distributive data storage part, the slave nodes run the HDFS partitions allocating free memory space for the data passed. The namenode maintains the status and the state of each slave node and their metadata, the secondary namenode is used to backup all the information from the namenode which in case of a failure of the master node can create a new node as the master node.

CHAPTER 3

PROMINENT STREAKS DISCOVERY ON BLOG ARTICLES

3.1 Problem definition

The basic unit of information is a blog article. An article consists of subset of entities where e = $\{e_i \ldots e_j | e_i, e_j \in E\}$, where E is the set of all given entities. An entity can be an event, place, person, organization etc. The entity count c_i is represented as the number of occurrences of an entity in a blog article. An entity e_i would be present in the blog articles, a sequence of count c_e = $\langle c_e^1, c_e^2, c_e^3 \ldots c_e^n \rangle$ is generated over an interval of time. The count of an entity for a day, c_e^i = $\sum_{a \in A_i} c_{e,a}$ where $c_{e,a}$ is the count of an entity in an article and A_i = $\langle a \mid article\ posted\ in\ day\ i \rangle$. The sequence of count generated for a single entity is defined as entity sequence.

Consider an example, *On Sep 01,2008, Barack Obama gained popularity in his campaign from the last 15 days.*
Barack Obama is the entity identified in the above example and the count captured is 1. Furthermore, the summation all the count of Barack Obama for a particular day will lead to the c_i for a day. When the count over the last 15 days are found, the count sequence is generated and would be able to notice the gradual increase in the count of number of times the named entity was mentioned.

(Streak and Prominent Streak) Given an n-element sequence of counts F = $\langle F_1, F_2, F_3 \ldots F_n \rangle$, a streak is an interval-value pair $\langle [l, r], v \rangle$, where $1 \leq l \leq r \leq n$ and $v = min_{l \leq i \leq r} F_i$.

Consider two streaks $s_1 = \langle [l_1, r_1], v_1 \rangle$ and $s_2 = \langle [l_2, r_2], v_2 \rangle$. We say s_1 dominates s_2, denoted by $s_1 > s_2$ or $s_2 < s_1$, if $r_1 - l_1 \geq r_2 - l_2$ and $v_1 > v_2$, or $r_1 - l_1 > r_2 - l_2$ and $v_1 \geq v_2$. With regard to P = $\langle P_1, P_2, P_3 \ldots P_n \rangle$, the set of all possible streaks is denoted by

S_P. A streak $s \in S_P$ is a prominent streak if it is not dominated by any streak in S_P, i.e., $\nexists s'$ s.t. $s' \in S_P$ and $s' {>} s$. The set of all prominent streaks in P is denoted by PS_P .(from [5])

(Top-k Prominent Streak) With regard to a sequence $P{=}(p_1, ...p_n)$ and its local prominent streaks $\mathcal{LPS}_P$, a streak $s{\in}\mathcal{LPS}_P$ is a *top-k prominent streak* if it is not dominated by k or more streaks in $\mathcal{LPS}_P$, i.e., $\big|\{s'|s' \in\mathcal{LPS}_P \text{ and } s'{\succ}s\}\big|{<}k$. The set of all top-$k$ prominent streaks in P is denoted by $\mathcal{KPS}_P$. Note that there can be more than k top-k prominent streaks.

Top-*k* prominent streaks are those local prominent streaks dominated by less than k other local prominent streaks, by the above definition. This definition has two implications. First, a top-*k* prominent streak must be locally prominent. It does not qualify even if $k > 1$ and it is only dominated by 1 subsuming streak. Second, a streak can qualify even if it is dominated by k or more other streaks, as long as less than k of those dominating streaks are local prominent streaks. (from [6])

Consider a sequence $P{=}(10, 20, 15, 30, 5, 5, 15, 10, 15, 5)$, corresponding to the count made by a named entity. The streak $\langle[3, 4], 15\rangle$, though only dominated by $\langle[2, 4], 15\rangle$, is a sub-streak of the latter, and hence is not a top-2 prominent streak. The intuitive explanation is that, $\langle[3, 4], 15\rangle$ is within the interval of $\langle[2, 4], 15\rangle$, therefore it is not considered important. On the other hand, the streak $\langle[7, 9], 10\rangle$ is a top-2 prominent streak. Although it is dominated by 3 streaks $\langle[1, 4], 10\rangle$, $\langle[1, 3], 10\rangle$, and $\langle[2, 4], 15\rangle$, the dominating streaks are all from the same period and only 1 of the 3 is a local prominent streak.

(Multi-sequence Prominent Streak) Given multiple sequences $\mathcal{P}{=}\{P^1, \ldots, P^m\}$ and their corresponding sets of streaks $\mathcal{S}_{P^1}$, ..., $\mathcal{S}_{P^m}$, a streak $s \in \mathcal{S}_{P^i}$ is a *multi-sequence prominent streak* in $\mathcal{P}$ if there does not exist a streak in any sequence that dominates s. More formally, $\nexists s', j$ s.t. $s'{\in}\mathcal{S}_{P^j}$, and $s'{\succ}s$. The set of all multi-sequence prominent streaks with regard to $\mathcal{P}$ is $\mathcal{PS}_{\mathcal{P}}$. (from [6])

As an example, consider 3 sequences corresponding to the points made by 3 named entities $P_1 = (22, 35, 25, 35, 3, 3, 12, 11, 15, 5)$, $P_2 = (11, 5, 30, 35, 20, 25, 2, 15, 2, 25)$ and $P_3 = (5, 10, 15, 5, 25, 10, 20, 5, 15, 10)$. The streak $\langle [1, 4], 22 \rangle$ of P_1 is a prominent streak within P_1 itself, but is dominated by $\langle [3, 6], 20 \rangle$ in P_2. Hence it is not a multi-sequence prominent streak.

CHAPTER 4

TECHNIQUES

4.1 Summary of techniques

Due to the massive size of the data, the proposed solution is a distributed data processing framework called MapReduce, particularly Hadoop which is one of its open-source implementations. Blog articles are passed as input to the mapper and the count of occurrences for a specific entity is found. The count is aggregated with respect to the entity and the day on which the article was created. Hence, a sequence of counts over a specific interval of time representing an entity is generated for the entire blog article dataset.

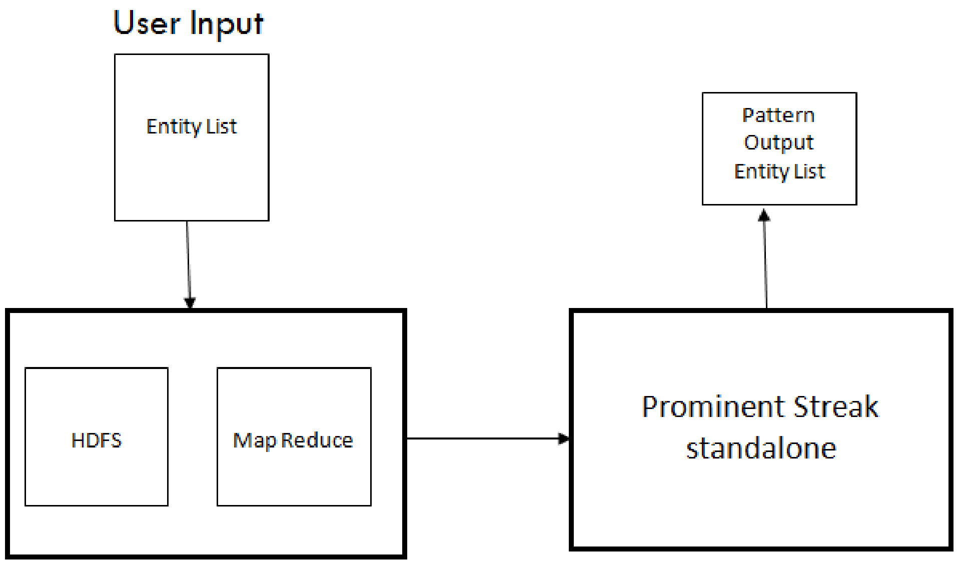

Figure 4.1. Workflow of the framework

Once the respective day and count is found, the next step being to analyze the data and find meaningful patterns. The prominent streak algorithm is applied over the count sequence for a specific entity to find meaningful patterns. Comparing between similar entities using the Multisequence prominent streak algorithm to find the entity which dominates all the other comparable entities for a particular interval of time. Finally, the Top K Multisequence prominent streak algorithm helps to keep track of the list of top k prominent entities from a given entity list. Figure 4.1 shows a representation of the scalable framework that is proposed.

4.2 Applying MapReduce framework to find sequence data

A distributed framework is considered for processing large magnitudes of unstructured data and the ideal choice would be Hadoop which is an open source implementation of MapReduce. Hadoop helps in managing large forms of key-value pair and batch processing on large scale data. In our MapReduce job, the input to the mapper is a single blog article with its respective date and a list of entities which are entered by the user. The output of the mapper are the entities with its respective date and the count of occurrences of that entity. The list of entities passed to the mappers globally is sorted. In the reducer phase, the entities are grouped by the entity label and the date which leads to finding the total count of an entity on a particular day. In the next MapReduce job, the input to the mapper was the dates and entities with the count and emits the entity labels with their dates and count together. In the reducer phase, we group by the entity labels and form the sequence of count for the specified interval of time. The two approaches proposed for the above methods techniques are brute force (Hashing) approach and the trie-based data structure approach.

The brute force (Hashing) method takes the input as a single blog for a day with the particular date, the list of entities are stored using a Hashset and passed globally to each mapper. In this technique, every entity is compared with an article in the blog using exact

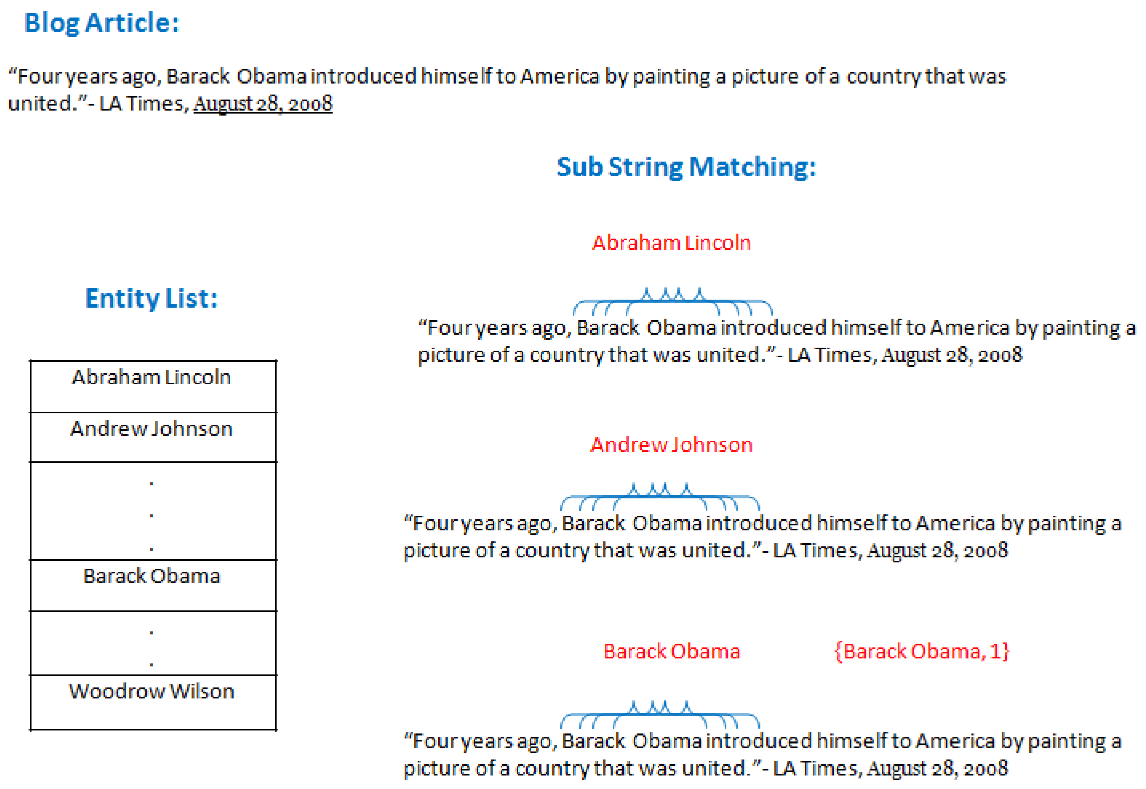

Figure 4.2. Hashed Entity List compared with Blog article

substring matching and when an occurrence is discovered, the count is incremented. The entity and its date with its corresponding count is passed to the reducer as shown in the figure 4.2. The reducer groups the entities by the entity label and dates giving the total count of a specific entity on a particular day. Algorithm 1 explains the working of the substring matching in mapper and the reducer algorithm is explained in Algorithm 2.

Since the complexity function tends to be in exponential order for the above approach which led to an intuitive approach of using a trie-based data structure. The entity list is saved in the form of a trie data structure such as an entity label "Barack obama" is saved

Algorithm 1: Mapper algorithm for entity discovery using sub string matching

```
Input: $BLG$: Blog Articles ; $EQlist$: List of sorted Entities;
Output: Entity + Day; frequency of the article
1 foreach $EN \in EQlist$ do
2     if $EN \in BLG$ then
3         $FN \leftarrow count(EN)$ // Counts the number of times the entity exists in
              the blog article.
4         $emit(EN, FN)$ // Sends to the reducer the entity and frequency.
5     end
6 end
7 return Entity ; Day + Total frequency
```

Algorithm 2: Reducer algorithm for entity discovery

```
Input: Entity+Day, frequency of the article
Output: Entity+Day, Sum of frequency of the article for a day
1 $Sum \Leftarrow Sum + value$; // Blog Article is split into characters.
2 return Entity+Day, Sum of frequency of the article
```

as ‘B’ as the parent node and its child node being ‘A’ and follows adding each character to the child list as shown in figure 4.3.

Consider an entity label name such as “Barack Husein” is made after the entity “Barack Obama” is created, the trie uses binary search to find if the character ‘B’ exists in the parent entity list. Since ‘B’is present due to the creation of the previous entity “Barack Obama”, the character ‘B’ is not added to the trie list of entities. The same process of binary searching is applied to its children for the entity label ‘A’, ‘R’, ‘A’, ‘C’, ‘K’,‘ ’. When the letter ‘H’ is searched in the trie data structure, the only child existing in the set

Algorithm 3: Mapper algorithm for entity discovery using trie data structure

```
Input: $BLG$: Blog Articles ; $EQlist$: List of sorted Entities;
Output: Entity + Day; frequency of the article
// Initialize data structures
1  $LBLG \leftarrow BLG.tochar()$ // Blog Article is split into characters.
2  $ChildList \leftarrow \{\}$
3  foreach $FN \in LBLG$ do
       // Considering each character as a node.
4      if $!ChildList.empty()$ then
5          $CN \leftarrow ChildList$ // For every child in the in the existing children list.
6          foreach $CN \in ChildList$ do
7              if $FN \in CN$ then
8                  $update(ChildList \leftarrow FN)$// Updates the existing child with
                       the added node in the children list.
9              end
10             else
11                 $ChildList.remove(CN)$// Remove the existing child from
                       the children list.
12                 $print(CN)$// emits the word with a frequency 1 to the reducer.
13             end
14         end
15     end
16     if $FN \in EQlist$ then
17         $ChildList \leftarrow FN$
18     end
19 end
20 return Entity+Day ; frequency of the article
```

Entity List:

Abraham Lincoln
Andrew Johnson
. . .
Barack Obama
. .
Woodrow Wilson

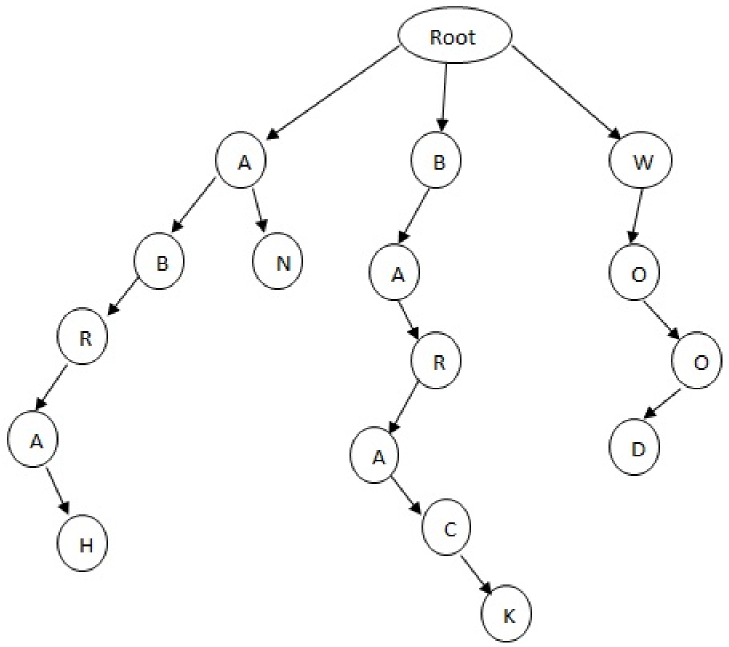

Figure 4.3. Trie data Structure with Entity List

Blog Article:

“Four years ago, Barack Obama introduced himself to America by painting a picture of a country that was united.”- LA Times, August 28, 2008

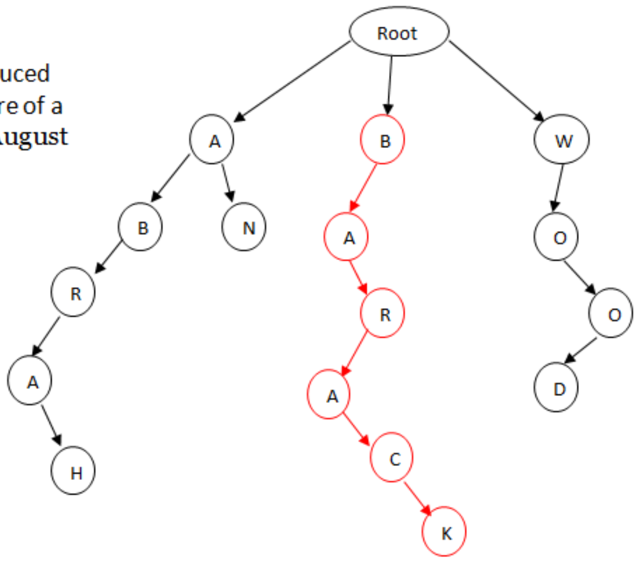

Figure 4.4. Trie data Structure compared with Blog article

of child nodes is ‘B’ and ‘H’ is not present. Hence, a new child is added after the list of ‘B’, ‘A’, ‘R’, ‘A’, ‘C’, ‘K’, ‘ ’ entity labels named ‘H’. Thus, the entity list is build with the trie-based data structure and passed globally to each mapper. In the mapper, each character in the input data is searched whether it is present in the entity list or not using binary search. If a final character of an entity is encountered, then the count is emitted to the reducer as shown in figure 4.4. The working is shown in algorithm 3. The reducer groups the entities by the dates and entity label finding the total count of a specific entity on a particular day as explained in Algorithm 2.

Algorithm 4: Mapper algorithm for sequence generation

```
Input: $ENDAY$: Entity+Day ; Total frequency
Output: Entity; Day+Total frequency
1 foreach $EN \in ENDAY$ do
2     $Split(EN \Rightarrow Entity, Day)$ //  Split the combination of Entity and day
          into two seperate values.
3 end
4 return Entity ; Day + Total frequency
```

The brute force method (Hashing) approach takes the list of entities as the input and compares each entity with a blog article. Using this method, the performance decreases quite dramatically as the number of entities grow larger. An example of the problem presented, given a list of 1000 entities the time taken to execute the job was 12 hours where as for a number of 2000 entities, it took a time of of 23 hours to complete its execution on a single machine. Due to the issue in performance, we tailored a data structure based on trie which helps to improve the runtime very efficiently. For the above example, using

Algorithm 5: Reducer algorithm for sequence generation

```
Input: Entity; $FREQDAY$: Day+Total frequency
Output: Entity, Sequence of frequency for specified interval of time
// Initialize data structures
1 $Date \leftarrow DateRange()$ // Specified date interval.
2 $CN \leftarrow \{\}$
3 foreach $FREQ \in FREQDAY$ do
4     $CN \leftarrow Parse(FREQ, Date)$ // extracting the frequency in the order of
          the provided date.
5 end
6 return Entity , Sequence of frequency for specified interval of timee
```

a trie-based data structure for a given list of 2000 entities, the time taken to complete the execution on a single machine has drastically come down to 4 hours.

Using the output of the MapReduce job, a series of entities with their dates and count of their occurrences as an input for the new MapReduce job, the mapper phase takes the input and emits the key as the entity label and the value as date and count. In the reducer phase, the values are grouped with respect to the entity label and a sequence of value is generated in the order of the days. The Algorithms 4 and 5 shows the working of the mapper and reducer.

4.3 Applying prominent streaks algorithm

When the count sequence for each entity is formed, the prominent streak algorithm is introduced. Applying the linear local prominent streaks (LLPS) method from the paper [7], which guarantees to produce a linear number of candidate streaks even in the worst case.

This method iterates through the sequence of data one day at a time and computes the candidate streaks, each candidate streak is maintained in a binary search tree (BST) to find the local prominent streak (LPS). If a candidate streak is dominated by a new sequence value, then the candidate streak is removed from the BST else the interval range is updated for the existing candidate streak in the BST. Hence, the prominent streaks are the streaks which are not dominated by any other streaks for an entity over an interval of time. With regard to the baseline method and NLPS, they may produce candidates that are not local prominent streaks. A candidate must be pruned if another candidate streak dominates it and subsumes it. (Note that they both produce candidates with the same right-end of interval at the same time. Therefore a candidate cannot be locally dominated by existing points in the current skyline.) The LLPS method helps in identifying useful patterns such as to find if a sudden burst exists in the sequence or not and if a particular entity has any identifiable patterns in it.

The extension from single-sequence algorithms (LLPS) to multi-sequence algorithms is intuitive. We process individual sequences one by one by a single-sequence algorithm and use a common dynamic skyline to maintain their prominent streaks. When a local prominent streak within a sequence P_i is identified using the LLPS, it is compared with current streaks in the dynamic skyline which contains prominent streaks from all sequences.

The top-*k* multi sequence prominent streak algorithm helps to keep track with the top-*k* list of entities which were dominated. For LLPS, since the candidates produced are guaranteed to be local prominent streaks only, we simply need to maintain a counter for each current skyline point in the dynamic skyline. The counter of a point records the number of its dominators in the skyline. When a candidate is compared against current skyline points, it is inserted into the skyline if it has less than k dominators. A current skyline point is removed if its counter reaches more than k.

CHAPTER 5

EXPERIMENTS AND EVALUATION

5.1 Setup

The experiments were conducted on 4 computers, each with 2.0 GHz Duo Quad Core Xeon E5405 under Ubuntu 10.04. The limit on the heap size of Java Virtual Machine (JVM) was default set to 1024 MB. The MapReduce framework used was fine tuned to support a mapper job to every core on the distributed system. The implementation of a trie-based data structure in MapReduce requires higher java heap space as the number of entities increases, since the trie data structure is preserved in the memory.

The dataset used for this research is from ICWSM conference provided by Spinn3r.com is a set of 44 million blog posts made between August 1st and October 1st, 2008 [8]. The post includes the text as syndicated, as well as metadata such as the blog's homepage, timestamps, etc. The data is formatted in XML and is further arranged into tiers approximating to some degree search engine ranking. The total size of the dataset is 142 GB uncompressed, (27 GB compressed). The data is cleaned up as various languages/locale blogs existed in the same data set, the set of blog which are considered are English. Furthermore, every line in the file is a blog article which is to be processed in a map reduce framework without the XML Tags.

The list of wikipedia entities is derived from the paper [9] which classifies the wikipedia entities into different categories. These entities helps us identifying which categories can be grouped together to find prominent streaks and get meaningful information from the blog dataset. A list of the top 10 categories which were used for the experiments and the count of the number of entities for their respective categories are shown in the table 5.1. The

Category Type	Number of Entities
nba season	242
film	30509
scientist	4673
artist	1165
disease	3624
writer	4601
actor	3770
person	12344
nba player	1422
political party	124

Table 5.1. Top 10 categories used and their respective entity count

total number of named entities involved in the wikipedia classification were over 500,000 entities and classified into 337 categories.

5.2 Results and comparisons

The charts below shows the performance between our hashing based technique versus the trie-based data structure technique using hadoop with 6, 12 and 24 mappers in figures 5.1, 5.2, 5.3 respectively and figure 5.4 shows the comparison between all the mappers considered.

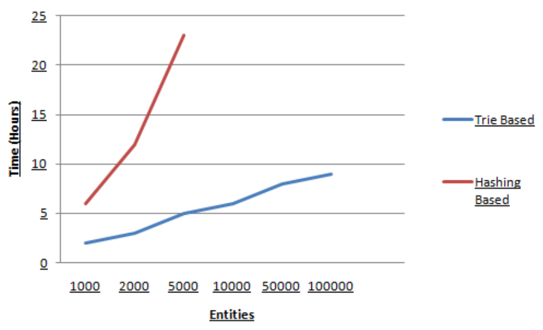

Figure 5.1. Performance with 6 mappers

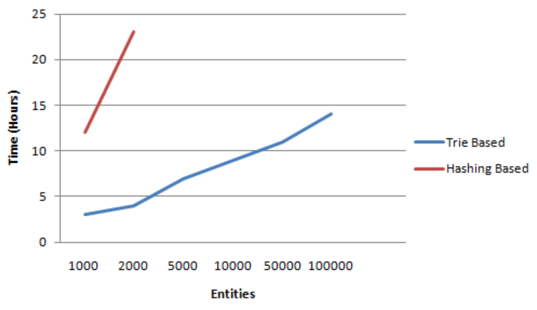

Figure 5.2. Performance with 12 mappers

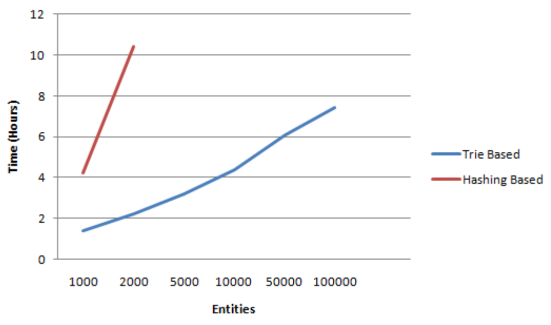

Figure 5.3. Performance with 24 mappers

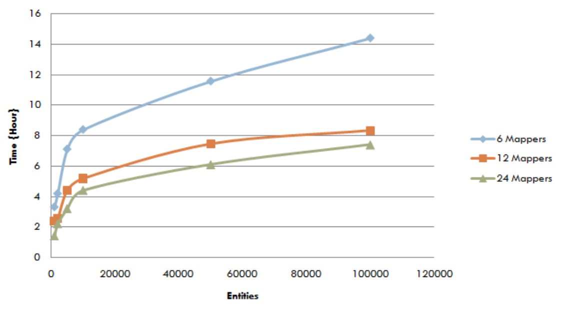

Figure 5.4. Performance of mutiple mappers

A few of the examples of interesting stories discovered using prominent streaks algorithm for various categories are shown below:

Politicians

- Among all the politicians in the time interval of 10th August - 31st September, 2008, we find that *Barack Obama* stood out quite prominently in terms of popularity having being talked about at least 3215 times everyday and the reason being the impending elections in 2008 which he won.

Actors

- We notice that on *August 9th 2008, the most talked about actor was Bernie Mac* with a count of 7328 times which corresponds to the real event of the death of *Bernie Mac* when he was 50 years old making him the most popular actor among all the other actors.
- Another interesting fact, we noticed was that *Tina Fey* was talked about at least 114 times everyday from September 13th to the end of the month due to her appearance on Saturday Night Live show where she impersonated the vice presidential candidate "*Sarah Palin*".

Basketball Players

- On the 8th of August, *Yao Ming* was the most talked about basketball player among all the other players of about 612 times on that particular day which when referred to real life events, we discovered the article "Yao Ming leads Team China into opening ceremony on August 8th".
- "*Kevin Duckworth* was an American professional basketball player who played for the nba died on August 25th, 2008". We found the pattern as he was the most popular basketball player having being talked about at least 1183 times a day for the time interval of August 26th, 2008 to September 3rd, 2008.

Diseases

- The most talked about diseases in the world in the two months of August and September, 2008 was *Cancer* having a count of 3703 times and the second talked about disease was *Anthrax* having a count of 3593 times and the third was *pneumonia* having a count of 1749 times.

CHAPTER 6

CONCLUSION

A scalable framework which can analyze and discover interesting stories from unstructured dataset has been designed. A trie-based data structure was used to efficiently discover named entities with the help of Hadoop which makes our model scalable and efficient at the same time. Applied the prominent streak algorithm to a list of entities count sequence to discover whether there exists an interesting story or not. Finally, the multisequence and top k multisequence prominent streak algorithm was used to compare similar entities and keep track of top-*k* prominent ones.

REFERENCES

[1] J. H. Kietzmann, K. Hermkens, I. P. McCarthy, and B. S. Silvestre, "Social media? get serious! understanding the functional building blocks of social media," *Business Horizons*, vol. 54, no. 3, pp. 241 – 251, 2011, special issue: social media. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0007681311000061

[2] A. Kotov, C. Zhai, and R. Sproat, "Mining named entities with temporally correlated bursts from multilingual web news streams," in *WSDM*, 2011, pp. 237–246.

[3] M. Wang, N. H. Chan, S. Papadimitriou, C. Faloutsos, and T. M. Madhyastha, "Data mining meets performance evaluation: Fast algorithms for modeling bursty traffic," in *ICDE*, 2002, pp. 507–516.

[4] T. White, *Hadoop: The Definitive Guide*, 1st ed., M. Loukides, Ed. O'Reilly, june 2009. [Online]. Available: http://oreilly.com/catalog/9780596521981

[5] X. Jiang, C. Li, P. Luo, M. Wang, and Y. Yu, "Prominent streak discovery in sequence data," in *KDD*, C. Apté, J. Ghosh, and P. Smyth, Eds. ACM, 2011, pp. 1280–1288.

[6] C. Li, X. Jiang, P. Luo, G. Zhang, and M. Wang, "Discovering general prominent streaks in sequence data," December 2012, work in progress.

[7] C. Apté, J. Ghosh, and P. Smyth, Eds., *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*. ACM, 2011.

[8] K. Burton, A. Java, and I. Soboroff, "The icwsm 2009 spinn3r dataset," May 2009, san Jose.

[9] A. Sultana, Q. M. Hasan, A. K. Biswas, S. Das, H. Rahman, C. Ding, and C. Li, "Infobox suggestion for wikipedia entities," in *CIKM*, X. wen Chen, G. Lebanon, H. Wang, and M. J. Zaki, Eds. ACM, 2012, pp. 2307–2310.

BIOGRAPHICAL STATEMENT

Jijo John Philip received his B.S. degree in Electronics and Communications Engineering from Visvesvaraya Technological University, India, in 2006 and his M.S. degrees from The University of Texas at Arlington in 2012 in Computer Science and Engineering in Fall 2012. His current research interest is in the area of Social Media analysis and large scale distributed databases. He is a member of IEEE society.