



NTNU – Trondheim
Norwegian University of
Science and Technology

Event Detection using Wikipedia

Eirik Emanuelsen
Martin Rudi Holaker

Master of Science in Computer Science

Submission date: June 2013

Supervisor: Kjetil Nørvåg, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Task Description

Analysing huge text based corpora has been a popular field for research, and recently Wikipedia has become a popular corpus to study. This project aims to make a scalable system with web interface for exploring the Wikipedia page view statistics combined with page edit statistics and try to detect popularity spikes. The system can be extended with other features, for example exploring relations and grouping of entities.

Preface

This is the final thesis of our Master's degree from the Norwegian University of Science and Technology, Department of Computer and Information Science. It has been prepared over a period of 20 weeks during the spring of 2013.

This thesis is within the field of Data and Information Management, and presents a system capable of performing event detection on Wikipedia.

We are very grateful to our two academic supervisors; Professor Kjetil Nørkvåg (NTNU) and Associate Professor Krisztian Balog (University in Stavanger). During the development of the thesis you have provided guidance that has been invaluable for the final result. Thank you very much for your support, advisory and involvement in the process.

Martin Rudi Holaker and Eirik Emanuelsen

Trondheim, June 2013

Abstract

The amount of information on the web is ever growing, and analysing and making sense of this is difficult without specialised tools. The concept of Big Data is the field of processing and storing these enormous quantities of data. Wikipedia is an example of a source that can benefit from improvements provided by this concept. It is an online, user driven encyclopedia that contains vast amounts of information on nearly all aspects known to man. Every hour, Wikipedia releases logs showing the number of views each page had, and it is also possible to gain access to all edits as Wikipedia frequently release the entire encyclopedia - containing all revisions. This makes it a great source when studying trends of recent years. In order to systematise these page views and edits, we design a scalable database, and implement a number of analysing jobs to process this information. From the page views and edit count, we perform burst and event detection, and compare the usefulness of two methods used for this purpose. We test the validity of our system by examining the case of the of the football transfer window in August 2011. Our system generate admirable and accurate results in regards to being able to detect these football transfers as events. In order to visualise the information we gather, we design a web application that give users structured access to our findings.

Sammendrag

Mengden av informasjon på nettet øker konstant, og å analysere og hente mening fra dette er vanskelig uten spesialiserte verktøy. Big Data er fagfeltet for å prosessere og lagre slike enorme mengder med data. Wikipedia er et eksempel på en kilde som kan dra nytte av ny utvikling innenfor Big Data. Wikipedia er et brukerdrevet encyklopedi som inneholder store mengder informasjon om nær alle konsepter menneskeheten har utforsket og har kjennskap til. Hver eneste time tilgjengeligjør Wikipedia logger over antall sidebesøk hver Wikipediaside har hatt, og det er mulig å få tilgang til alle revisjoner ettersom Wikipedia med jevne mellomrom slipper hele encyklopediet - inkludert alle forhenværende versjoner. Dette gjør at Wikipedia er en fremragende kilde om man ønsker å studere nyere års trender. For å systematisere statistikken over sidevisninger og revisjoner modellerer vi en skalerbar database, og implementerer flere analyserende programmer for å prosessere denne informasjonen. Denne statistikken bruker vi til å finne eksplosjoner i sidevisninger og hendelser, og sammenligner to metoder for å beskrive og finne slike eksplosjoner og hendelser. Hvorvidt metodene våre er legitime bekrefter vi ved å se på overgangsvinduet i fotball i august 2011. Dette studiet viser at systemet greier å finne hendelser i form av fotballoverganger med stor presisjon. For å visualisere all informasjonen vi innhenter, lager vi en nettapplikasjon som tillater brukere å få tilgang til den på en strukturert måte.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Our Contribution	2
1.3	Introduction to the System	3
1.4	Challenging Parts	3
1.5	Scope of Project	4
1.6	Readers Guide	5
2	Related Work	7
2.1	Burst and Event Detection Methods	7
2.2	First Story Detection	8
2.3	Utilisation of Corpora	8
3	Pre-studies	11
3.1	Definition of a Burst	11
3.2	Burst Calculation Opportunities	12
3.3	Outlier Detection	13
3.4	Definition of an Event	15
3.5	Distinguishing Events from Bursts	15
3.6	Jaccard Coefficient	16
3.7	Granularity of the System	16
4	Technology Introduction	19
4.1	Storage Choice	19
4.2	HBase	20
4.3	MapReduce	22
4.4	DBpedia	24
5	HBase Table Design	27
5.1	Pagecount Table	27
5.2	Burst and Event Tables	28
5.3	Redirect and Related Entities Tables	31
5.4	Jaccard Tables	31
6	Burst and Event Detection	33
6.1	Sliding Window Implementation	33
6.2	Burst and Event Detection	35

7	Back End	39
7.1	Adding Page View Statistics	39
7.2	Edit Count	41
7.3	Calculating Jaccard Coefficient	41
7.4	Related Entities	42
7.5	Redirects	43
8	Front End	45
8.1	Search for Events and Bursts	46
8.2	Search for Entity	47
8.3	Jaccard Similarity Search	53
9	Examples	55
9.1	Burst Examples	55
9.2	Event Examples	59
9.3	Related Events and Bursts	63
9.4	No Detection Examples	64
9.5	Jaccard Similarity	64
9.6	Grouping of Entities	66
10	Results and Analysis	69
10.1	Football Transfers	69
10.2	Event Detection	76
10.3	Related Entities	76
10.4	Jaccard Similarity	76
10.5	Grouping Entities	77
10.6	Z-scores	77
10.7	Modified Z-scores	77
10.8	Modified Z-scores Versus Z-scores	78
10.9	Summary	80
11	Further Work	83
12	Conclusion	85
	Bibliography	87
	Appendices	
A	List of Transfers	93
A.1	BBC List	93
A.2	Refined List	99
B	Google Search	103
B.1	Bursts	103
B.2	Events	106
B.3	Related Bursts and Events	109

Chapter 1

Introduction

In this chapter we introduce our motivation for embarking on this thesis, we give a short introduction to the features and possibilities our system presents, as well as mention some of the challenges encountered during the project.

1.1 Motivation

Wikipedia consist of millions of articles with millions of visits each day. To utilise this information for something useful is an interesting challenge. Since this encyclopedia is community driven it also produces an enormous amount of edits. Analysing these statistics can give an accurate indication of the interest in topics and entities, and we believe it can be used to discover trends and events without any further, external resources.

In this thesis, we propose an application which is able to monitor the Wikipedia page views and page edit count. These data can be used to detect sudden spikes in the usage of a page, which is usually a sign of something happening. If only the page view data has a spike, it may indicate any number of trivial occurrences, such as being linked to from a newspaper article, or it being the entity in questions birthday. On the other hand when both the page views and the page edits experience a spike, it may indicate something more substantial, for example a president election, or a persons death. The goal is to extract historical data and be able to distinguish the difference between the aforementioned scenarios.

In order to build a system that manages to extract the above explained scenarios, we need to design a scalable database scheme, tailored for this very purpose. Our system will not implement updates, but the database will be constructed to be able to handle updates in the future. The challenges offered by this project are particularly interesting due to the fact that the required technologies are closely related to Big Data, the main interest field of both authors.

The way we want to test this system is to try to find football transfers. The reason for this choice is during a transfer window there exists an enormous amount of rumours which may lead to many page views of a rumour object page. The goal here is to manage to extract a high amount of transfers. It is also possible to look at time delays between Wikipedia and the actual announcement of a transfer, or if the system detects a rumour about a person before the actual

announcement. A challenge related to this is that the system uses daily and not hourly statistics. Another thing that can be interesting, is to see if there are explosions of edits only on the persons that have been sold to other clubs, or if Wikipedia users tend to edit players that have been the subject of rumours, players that have had a great performance, or similar.

We have decided to analyse the transfer window in August 2011. The reason for going so far back in time is the DBpedia provided files, which are dated late May / early June 2012¹. The 2012 January window could have been used, but there is usually more activity during the summer. In addition, all available data will not be used since the page view statistics takes up a lot of hard disk space and we are working with limited resources. The data that will be used is the page view statistics added at 0800 and 2000 from 1 August until 31 August, which is one twelfth of a monthly dataset. However, these two hours cover many time zones and we expect that these data will provide accurate and desired results.

Other Use Cases

There exists multiple scenarios in which a system like this would be most useful. Imagine a journalist contemplating what to write about in his next piece. He narrows down a few topics, but is unsure what to start on. He may then continue to look at the graphs for different Wikipedia pages related to some of the subjects he is considering, and from this, decide which article that is the most likely to be interesting for the broader audience. He may even discover that some of the topics have more interest in another time of the year - which convinces him to save that particular piece to a later time, when it probably will be more relevant - and thus generate more clicks for the newspaper.

Other uses may be to verify how up-to-date information is. If one check the statistics for an entity's Wikipedia Page and see that it has no significant edits in a very long time, there is ample reason to be sceptical to the contents of said Wikipedia page, and Wikipedia contributors may also want to add the page to their to-do list of pages to improve. Wikipedia moderators can also find usage in the system by verifying pages that have experienced a larger than normal amount of edits.

1.2 Our Contribution

This project have provided multiple contributions. First, we provide a HBase schema suitable to store Wikipedia page view and edit count statistics. We compare Z-scores against modified Z-scores for burst detection using a sliding window implementation. We have created a system that detects and distinguishes bursts and events with basis in Wikipedia page view statistics and edit count. The system is also able to detect relations between entities by applying the Jaccard Similarity Index on their burst-patterns. Finally, we have created a Web application that allows a user to browse Wikipedia page view and edit count statistics, as well as browsing identified

¹<http://wiki.dbpedia.org/Downloads38>

bursts and events, and finding related entities.

1.3 Introduction to the System

As mentioned, the system uses Wikipedia page view and page edit history statistics. We also utilise structured information from DBpedia (explained in Chapter 4). The page view statistics are usable straight from the source, but the edit count and the DBpedia files require some pre-processing. The system can extract bursts, events, Jaccard similarities and other statistics when given these files as input. The system has both a back end and a front end. The back end consists of a number of different MapReduce jobs that inserts the data into a highly scalable database, HBase. The front end uses the data from the HBase tables to show the user the data in a structured way. In the front end of the system the user has the opportunity to search for bursts and/or events on a desired date, bursts and/or events for specified entities, and Jaccard similarities for specified entities. In addition, the user can choose a burst level to filter out the bursts of a given level. The other option is to search for an entity, and if that entity exists, all the information about the entity contained in the system will be displayed. For a graphical overview of the system see Figure 1.1

1.4 Challenging Parts

One of the most challenging parts in this project was the creation of the various HBase tables. As we mainly have worked with SQL before, it was hard to adapt to a NoSQL database, and understand how to structure the tables. In addition, the Hadoop distribution we use - the Cloudera Manager Free - come with HBase 0.94. This version is not optimized for updating tables and can not scan multiple tables, which have had an impact on how we have structured the data store. For example we would have split some of the tables into two tables, one for each column family.

There exist many different methods for detecting bursts in data streams, and we had to choose which one to use, or decide whether to create our own approach or not. We first tried and failed with a couple of our own approaches, but after reading a lot about burst detection we decided to use a sliding window, and detect outliers in this with Z-scores.

Another challenging part was to set the different thresholds for Z-scores, explained in Chapter 3, and edit counts. We solved this a bit unlike to how other research projects solved this. For the Z-scores we set the threshold much higher than other research projects, which in many cases use 3-3.5 as a minimum Z-score. In this project the Z-score has to exceed 10 in order to get into the lowest bursting category. We felt that the 3.5 value was too low and gave us too many results that we did not see as outliers. With a minimum Z-score of 10 we only get the more extreme outliers. On the other side we could have achieved much better results with the usage of a minimum Z-score of 3.5, but this would have reduced the precision heavily.

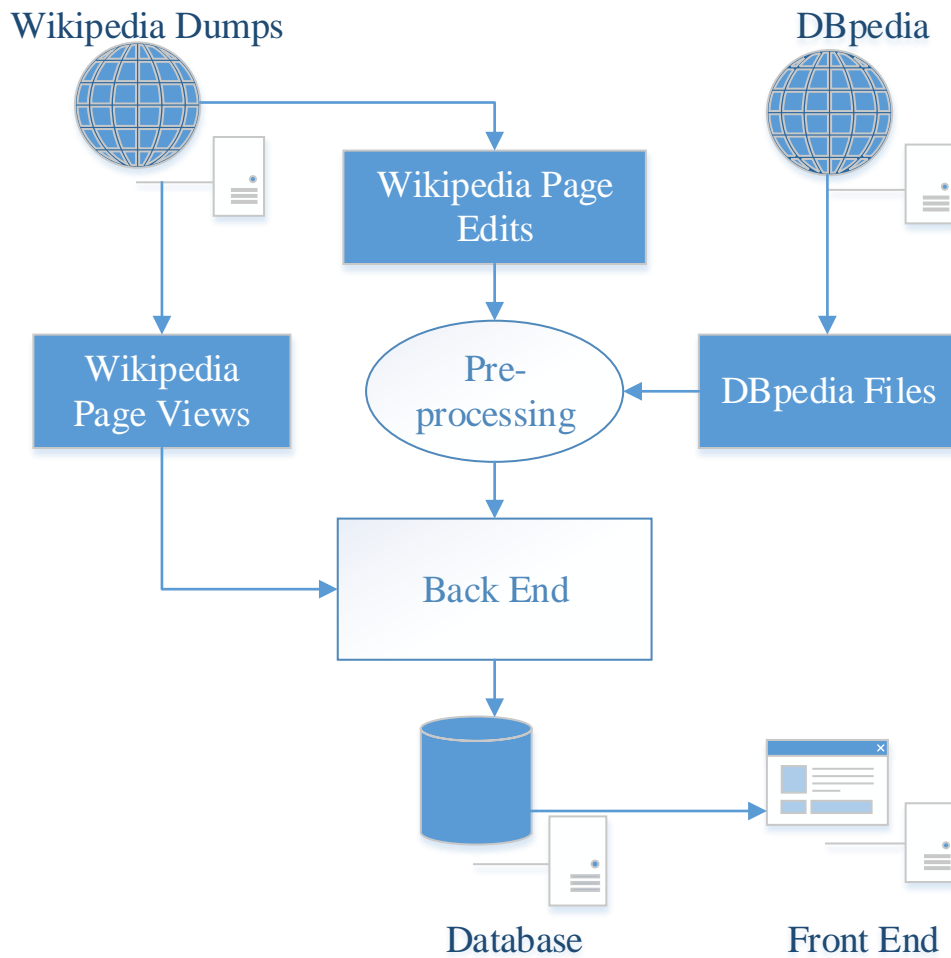


Figure 1.1: System Overview

1.5 Scope of Project

This project was accomplished over 20 weeks with limited material resources, mainly the authors laptops. A consequence of this was having to use a small data set, spanning over one month. However, the design of the database is optimized for scale out.

Requirements

In the following section we present our requirements for a successful system:

RQ1: A scalable database scheme

This is a crucial requirement for our system. As we will be working with the Wikipedia page view statistics, we need a database scheme capable of storing vast amounts of information. We

also need a database scheme that is able to handle data being continuously added.

RQ2: Add Wikipedia page view statistics and edit statistics to the database

Our system needs to be able to add all page view and edit statistics to the database.

RQ3: Detect bursts using these page view statistics

With the aforementioned page view statistics being added to the database, the system shall be able to detect bursts with these page view statistics as a source.

RQ4: Distinguish bursts from events using edit

While being able to detect bursts, our system should be distinguish events from these bursts through the usage of edit statistics.

RQ5: Find related entities

The system shall be able to find related entities. This shall be achieved through the usage of DBpedia files, as well as Jaccard similarities.

RQ6: Graphical User Interface

The system shall contain a Graphical User Interface (GUI), allowing the user to perform a magnitude of tasks. This include showing a graphical view of the data connected to an entity, allowing the user to search for entities in order to see their page view and edit counts. The user shall be able to see bursts and events for any given date. In addition, the user shall be able to search for Jaccard similarities. The process of searching for entities shall be improved with redirect for commonly misspelled entities. Any bursts and/or events for related entities shall be displayed to the user.

1.6 Readers Guide

In this section we will describe the general outline of the report, and briefly mention the content of each chapter.

Introduction and Pre-study

The first 4 chapters act as an introduction to both the report, and to important subjects in our work.

Chapter 1 contains the motivation for the project and a rough guide of what we want the system to accomplish.

Chapter 2 gives an overview of related work in regards to burst and event detection, different studies and systems that utilise these techniques.

Chapter 3 will further elaborate our approach to burst and event detection, how we conduct outlier detection, as well as the method we use for similarity grouping.

Chapter 4 introduces the technologies and resources crucial to the completion of the project.

Own Contributions

The next 4 chapters will describe our own work in great detail.

Chapter 5 describes how we design our HBase tables. We present the RowKey design for all tables, as well as the the column families and qualifiers. In addition, we provide example rows from all tables.

Chapter 6 shows our algorithms for burst and event detection, and describe how we add these to HBase.

Chapter 7 describes the various MapReduce jobs that add Wikipedia page view statistics and DBpedia files to HBase.

Chapter 8 describes the web application we use to present our results in detail.

Results and Conclusion

The last 4 Chapters present the results our system produce, further work, and the conclusion to the project.

Chapter 9 shows the functionality of the web application.

Chapter 10 presents how our system perform in a case study of the football transfer window in August 2011.

Chapter 11 presents some development possibilities for the system.

Chapter 12 concludes the report, and give our final remarks on the system and our results.

Chapter 2

Related Work

In this chapter we present different articles that focus on burst and event detection, as well as projects related to similar concepts - such as first story detection, and mining information from vast online data sets, such as social media and Wikipedia.

2.1 Burst and Event Detection Methods

According to [39], burst detection is the activity of finding 'abnormal aggregates in data streams'. Furthermore, a definition of an Event is given in [24] as 'something that happens at some specific time and place'.

Kleinberg [18] propose using State Models for burst detection, namely a Two State Model and an Infinite State Model. The states in these models are corresponding to the rate at which they are emitting messages, and bursts are signalled by changing from a lower to a higher state. It is possible to control and prevent short bursts by assigning a certain cost to changing states. Kleinberg's work has been utilised to for example detect Buzzwords from document streams [34], performing scalable and near real-time burst detection in eCommerce queries [22], and [36] can capture semantic and temporal information in a document set.

Windowed burst detection is introduced in [10, 11, 38]. A *window* can be described as having a length n , and consisting of values ranging from T_1, T_2, \dots, T_n . These windows can be further split down if desired. In [10], they propose a system that mines *systematically evolving data*, defined as data that changes through addition and deletion of record sets. Here they introduce an *unrestricted* window, also called *landmark* window, consisting of all data collected so far. They also propose the *most recent* window, which is of length n , consisting of the n last observed values. This is also called a *sliding* window. Gehrke et al. [11] have created a system that computes correlated aggregates over continual data streams. They tested the system using two different window methods, namely landmark window and sliding windows. For the sliding window they use a window size of 500 data points. [38] monitors thousands of data streams in real time. For a single stream they calculate statistics such as mean and standard deviation. They use a sliding window method where each sliding window contain multiple smaller windows. Fung et al. [9] propose an algorithm called Time Driven Documents-partition, that constructs an event hierarchy, based on a text corpus, relevant for the search string and date.

Elastic burst detection can be interpreted as a data set split into an amount of *windows*, and the size of the windows are dynamically chosen [27]. Zhu and Shasha [39] describe different viable window models, which can be used for elastic burst detection. Their main contributions are the Wavelet Tree, which is a window that is separated into smaller windows, and the Shifted Wavelet Tree, which is an elaboration of the Wavelet Tree. These trees are built up of levels, and the length of the wavelets increase as one moves closer to the top of the tree. This way of conducting burst detection has been adopted in other research projects like [14] and [13]. In [35], Zhang and Sasha explain problems concerning the Wavelet Trees and propose a new structure with a higher amount of levels, the Aggregation Pyramid. They also suggest using a heuristic for the determination of a static threshold. In [17] they build on much of what Zhang and Sasha presented, but also propose to forecast the values. When implementing this forecast, they use both a landmark window and a sliding window, together with *smoothing*. *Smoothing*, in their case, is the process of making sure that the most recent values are the most relevant for the next forecast.

2.2 First Story Detection

The concept of *First Story Detection* (FSD), also called *New Event Detection*, is to find the first story to discuss a particular event. The traditional way to perform FSD is to compare each new document to an ever growing set of documents. A naïve method, Nearest Neighbour [24], to perform FSD is to represent each document as a vector of terms, and comparing the similarity of any new vectors (documents) to all the other. If the similarity is below a certain threshold, a new First Story is detected. [24] propose a more advanced similarity measure for FSD on Twitter. They use a modified *locality sensitive hashing* method. While performing FSD with this method, each tweet is added to a thread, where threads contain tweets related to the same subject. Event Detection is then performed by looking at the growth rate of the threads. Osborne et al. [21] seek to improve First Story Detection on Twitter by using Wikipedia Page View files. As well as using a state-of-the-art FSD described by [24], they identify outlying Wikipedia Pages using a 48 hour sliding window and a Grubb's Test¹. They conclude that the usefulness of Wikipedia as a mean for real First Story Detection is limited due to latency, but that it greatly improves the chances of finding actual news when applied as a filter. In [28], an attempt to perform FSD on Wikipedia is described by utilising that all edits in Wikipedia are broadcast to Internet Relay Chat-servers (IRC-servers).

2.3 Utilisation of Corpora

Whiting and Alonso [32] present a system that uses Twitter hashtags to extract, events and map them to a maximum of three Wikipedia articles. For each hashtag from Twitter, they use occurrences of this hashtag and the page view statistics to find the best match. This project uses two page view files each day. [33] describe a system that generates *CrowdTiles* for websearches.

¹<http://www.itl.nist.gov/div898/handbook/eda/section3/eda35h1.htm>

These *CrowdTiles* consist of two parts. The first part contains Wikipedia articles which are related to the search theme and has been categorised as events. CrowdTiles classifies a Wikipedia article as an event if there has been more than 10 edits within the last 12 hours. The second part display tweets related to these articles. They use an API provided by Twitter to extract popular trends. Zhao et al. [37] aim to identify event-related bursts from Twitter. Tweets are treated as activity streams, whose probability is modelled using a Poisson distribution. During comparisons with for example [18], they conclude that smoothing is especially important in distinguishing events from bursts - at least when operating on tweets.

Ciglan and Nørvåg [6] describe WikiPop, a system² that allows the user to give a context as input, and receive information about related concepts that experience an increase in popularity. They identify these related concepts through processing the Wikipedia link graph, applying weight to the edges, and use the Spreading Activation algorithm on this weighted graph. Finding pages with increasing popularity is done by using Wikipedia page view statistics. [23] provide a RESTful interface which allows users to query Wikipedia access logs in various ways, such as the raw page view logs, and to find similar concepts and pages sharing trends and bursts. [29] use Wikipedia page view statistics from different languages versions of Wikipedia to analyse the usage patterns and trending patterns across the world. They also propose grouping entities using the categories inherent in Wikipedia create a baseline amount of expected page views, and analyse page views on a granularity of categories, as opposed to entities.

Georgescu et al. [14] aim to give people a comprehensive view of all aspects of an event, not only what ends up in the last version of a Wikipedia article. They examine the edit history of entities that experience an edit burst, extract event related updates, and further combine this with earlier bursts the entity has experienced. They use the Wikipedia Revision Toolkit [8] to get hold of the edit history. [13] checks the viability of using Wikipedia's edit history for extracting event-related updates. They perform an analysis of event-related updates in Wikipedia, such as the relation between edit spikes and events. They detect bursts using a version of [39]. The analysis of the edits include checking if the textual content of the edit contain a date which is in proximity to the current date, as well as semantically analysing the content of the edit to check for words indicating events, such as 'death', 'sold', and so forth. Finally, any tags attributed to the edits are also analysed, such as 'current', indicating current events, and 'rvv' - reverting vandalism, to name a few.

Balog et al. [2] use blogs as their dataset. They use only blogposts tagged with a mood. If a peak in the usage of a mood is detected, blogposts with this mood are analysed, and 'overused' words are extracted. These unusual words are then used as queries in a news corpus. In Identifying Similarities, Periodicities and Bursts for Online Search Queries [30], they create a time series from MSN search logs, with each query being treated like its own time-series. They compress the data and describe each time series using the k best Fourier-coefficients, making for efficient storage (at that time). They show that this way of storing time series maintain the patterns in the search logs. Further, they perform burst detection using a sliding window and x number of standard deviations approach, as well as comparing the similarities of bursts.

²<http://research.idi.ntnu.no/comidor/wikipop/>

Chapter 3

Pre-studies

In this chapter we present the definition of a burst and an event. In addition, various problems and challenges related to burst and event detection are further explored and explained, and we give our reasoning for our approach, and the techniques we utilise.

3.1 Definition of a Burst

In [39], burst detection is described as the activity of finding 'abnormal aggregates in data streams'. Additional words that can be used to describe bursts are peaks or explosions. In short, a burst is an abrupt increase in usage. In our case it can be seen as a peak in a graph showing traffic on a Wikipedia page. A peak is though not enough to mark a Wikipedia page as a bursting page, it has to have a significant deviation in a positive way from the average in order to be marked as bursting. In this project a burst only considers the number of Wikipedia page views. In Figure 3.1 there is an example graph that show what bursts will look like in a graph. The yellow line indicates the page views for an entity.



Figure 3.1: Bursty Line Chart Example

3.2 Burst Calculation Opportunities

Burst calculation has been presented in many different ways in related research projects [6, 18, 35, 37, 39]. In this section we will present those we chose to use in this project.

3.2.1 Threshold Based Burst Detection

Threshold based burst detection is mentioned in amongst others [17], and is a method where the detection of bursts is done by comparing a value against some threshold. The main challenges in this type of burst detection is finding the actual threshold with which to do the comparison. Low thresholds may result in low precision, with entries being wrongfully marked as bursts. Too high precision may lead to bad recall, risking that legit bursts are being discriminated away. Various methods and techniques in choosing thresholds, and finding what these thresholds will be applied to will be described in greater detail in the next section and in Chapter 6.

3.2.2 Windowed Burst Detection

Using windows in order to accomplish burst and event detection or other data mining operations is described in detail in amongst others [10, 11, 38, 39]. The three window methods described are in a varying degree suited to burst detection. A *landmark window* uses values from a given point to present, for example the average price of a CPU from release until present. In our case, the landmark could be 9 December 2007, the day Wikipedia started releasing page view statistics. This is also called an unrestricted window, a window consisting of all data values collected [10, 38]. A *sliding window* is using the last n days to calculate the value, for example the average price of a CPU over the last five days [10, 11, 38, 39]. For a graphical overview of Landmark and Sliding windows, see Figure 3.2. Damped window is a model where old data becomes less important compared to when they were new. For example, if keeping track of a running average avg_{new} , this score may be updated using the following formula (example from [39]):

$$avg_{new} = avg_{old} \times p + x + (1 - p), 0 < p < 1$$

In this project we will be using a Sliding Window implementation in favour of slightly more advanced wavelet or aggregated pyramids implementation [35, 39]. The main reason for this is that we have the knowledge that this approach has been used with success in previous work, for example in [21]. Reasons for not using a landmark window include that it is very rigid and slow to consider changes in the data set, as it tends to become very large [9]. However, a sliding window implementation in itself does nothing, one still needs some method to actually determine if an element is a burst, or an *outlier* in regards to the other elements in the window. This is the focus of the next section.

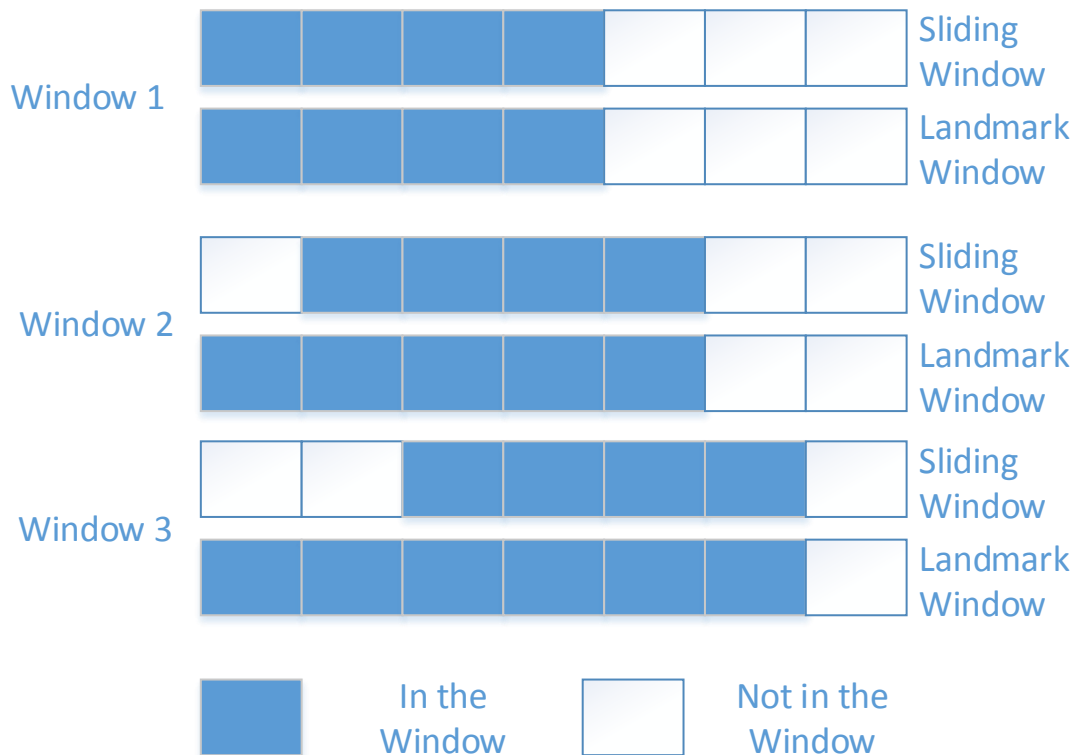


Figure 3.2: Landmark and Sliding Window Example

3.3 Outlier Detection

Here we present a few methods to detect outliers in a data set - in our case, a sliding window. It is worth to notice that both metrics we present work best under the assumption that the population they are used on are normally distributed, as well as that each individual window is being treated as an entire population, and not a sample. The reasons for not just setting a simple threshold, saying that any value $x\%$ larger than average is an outlier, or any value higher than y is an outlier, is that each data set and each entity is different. Such fixed thresholds do not yield good results [9].

3.3.1 Z-score

The Standard Score, or Z-score, is a simple test for indicating how many standard deviations a data point is deviating from the rest of the sample population. The formula for calculating the Z-score for an element in a population is as follows:

$$Z_i = \frac{(x_i - \mu)}{\sigma} \quad (3.1)$$

where

$$\text{Population} = x_1, x_2, \dots, x_n \quad (3.2)$$

$$\mu = \frac{\sum_{i=1}^n x_i}{n} \quad (3.3)$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \mu)^2}{n}} \quad (3.4)$$

If one is interested in positive burst - bursts where the x_i value is larger than the mean, one can disregard all results in which the result from Equation 3.1 is negative. As for the value of the Z-score, there are no definite solution as to what threshold should be used to denote an outlier. Regardless, an important challenge is to find appropriate value(s) for the input one will be operating on, as well as one that conforms to the results one wants. If the area of interest is only the heaviest outliers, a higher threshold will have to be chosen.

3.3.2 Modified Z-Score

Iglewicz and Hoaglin [16] propose using a Modified Z-score instead of the regular Z-score shown in Section 3.3.1. Their proposition is that one instead should use the following:

$$M_i = \frac{0.6745(x_i - \tilde{x})}{MAD} \quad (3.5)$$

where \tilde{x} is the median of the population (window in our case), and the Median Average Deviation (MAD) of the population is:

$$MAD = \text{median}(|x_i - \tilde{x}|) \quad (3.6)$$

This is performed on all elements in the population. As with the regular Z-score, there are challenges related to deciding what values that are to be classified as outliers. An example follows:

Population = 7, 13, 4, 2, 16
 Sort and Find Median → 2, 4, **7**, 13, 16
 Subtract Median → | - 5|, | - 3|, 0, 6, 9
 Sort and find MAD → 0, 3, **5**, 6, 9
 MAD = 5

3.4 Definition of an Event

Petrović et al. [24] describes an event as 'something that happens at some specific time and place', like the president election in the USA, Superbowl, or football transfers.

In this project two premises have to be fulfilled to categorize an entity as an event at a certain time. The first premise for an event is that the given entity is bursting, with the bursting aspect being explained above in Sections 3.2 and 3.3. The second premise is that the entity has been revised more than a fixed number of times. In figure 3.3 we show a graph that gives an impression of how we want to interpret an event. The yellow line marks the page views and the red line marks edits to the page.

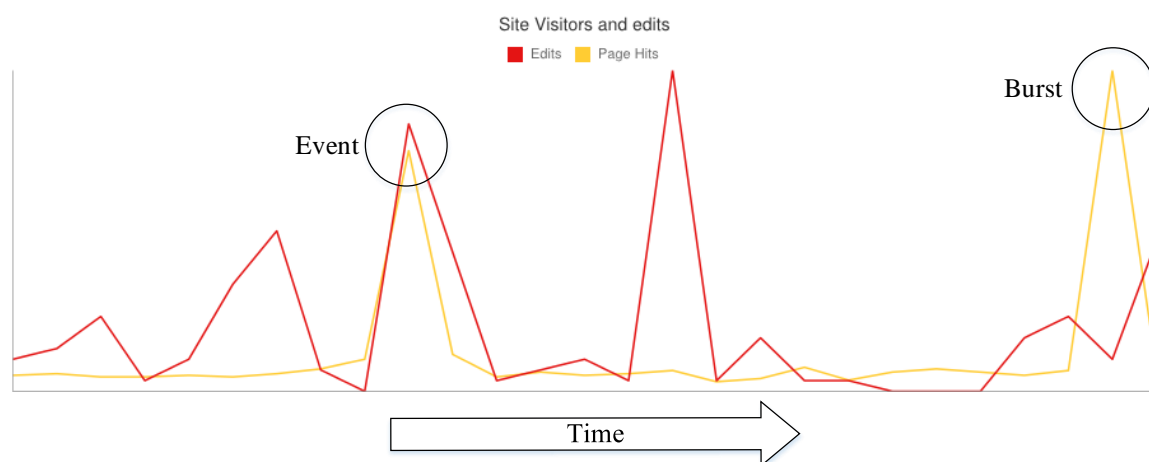


Figure 3.3: Event Line Chart Example

3.5 Distinguishing Events from Bursts

While the methods being mentioned above make it possible to detect bursts, they do nothing more than to denote the existence of outliers in a population. They do not explain the significance of the outlier, only the magnitude, although one might argue that these are two sides of the same coin. If the magnitude of an outlier is great enough, it is hard to imagine that it is insignificant. However, this would be an unreliable method with which to discriminate bursts from events.

An alternative is to utilise the Wikipedia Edit History. Each time a Wikipedia article is changed, much information is stored. This includes timestamp, who made the change, what was changed, the previous revision, and so forth. This opens up a lot of opportunities. Semantically analysing the contents of the edits can be very useful, for example checking for words indicating events, such as 'death', 'sold' and 'bought' [13]. This also opens up the possibility to say something about the significance of the edits, as edits can be tagged as minor - which would be appropriate

if the edit did nothing but change a spelling error - and edits can be tagged with "rvv" - reverting vandalism - which is used when false information is inserted into an article [13]. A simpler way of using the Wikipedia Edits is to look at numbers alone - how many edits have been performed on an article in a certain time period, such as in [23], where an event is required to have more than 10 changes in an article in the last 12 hours.

3.6 Jaccard Coefficient

The Jaccard Index, or Jaccard Similarity Coefficient, is a method for comparing the similarity of sets. The formula for calculating the Jaccard Similarity Coefficient is as follows [5]:

$$J_{X,Y} = \frac{|X \cap Y|}{|X \cup Y|} \quad (3.7)$$

Slightly more specific, if \mathbf{X} and \mathbf{Y} are both bit vectors of length n , the formula looks like the following:

$$J_{\mathbf{X},\mathbf{Y}} = \frac{\sum_{i=1}^n v_{11}}{\sum_{i=1}^n v_{11} + \sum_{i=1}^n v_{10} + \sum_{i=1}^n v_{01}} \quad (3.8)$$

where v_{11} denote that the bit is 1 at the same position in both \mathbf{X} and \mathbf{Y} , while v_{10} and v_{01} denote that the bit is 1 in the one vector, but not in the other - the *xor* function. $J_{\mathbf{X},\mathbf{Y}}$ will always be a number between 0 and 1, with a higher number meaning higher similarity. The bit vectors do not have to be of the same length in order to calculate the Jaccard Similarity Coefficient, but it does require that you always compare the same element, document, index or day - dependant on the nature of the sets you do the comparison on. In our case, an interesting factor would be to transform the burst patterns of entities to bit vectors, with 1 denoting a burst on a day, and calculate the value of Equation 3.8 between all of them. The idea behind this will be to find - perhaps unexpected - relations between entities and their burst patterns.

3.7 Granularity of the System

The page count files from Wikipedia are released with page click information down to the hour, so there is no problem in portraying the page count information on such a small granularity. The question is whether it is useful to utilise this level of detail. Even though hourly statistics are released, there is a delay of several hours from the hour they portray, to the release of the file. This slightly diminishes the usefulness of Wikipedia as the single source for automated, real-time *First Story Detection* or Event detection, at least if one is only utilising the page view dumps. A better alternative for this would perhaps be in conjuncture with the domain of social media, such as with Twitter [21, 28, 37]. By using Wikipedias IRC live updates of edits, Steiner et al. [28] have managed to create a system¹ that manages to detect events on a near real-time

¹<http://wikipedia-irc.herokuapp.com/>

basis - but even here, candidates for "breaking news" are cross-checked with social media sites to improve accuracy.

Despite this, it is not to say that being able to investigate page counts down to the hour is useless. If only looking at page view statistics of entire days, one might very well lose insight into the nature of a peak in views. If all the the extra page views generating a peak stem from a time span of only one hour, it is less likely that the peak is implicating an event. True events will likely generate interest for a longer time period [37], and one would likely need some extra information to distinguish these thought-to-be insignificant bursts from actual events. The edit history, and number of edits, may well be this form of information. If there is a big spike in page views, but no new edits have been added, it is reasonable to assume that nothing significant has happened to the spiking entity.

Chapter 4

Technology Introduction

In this chapter we introduce technologies and resources we will use in the project. We also elaborate on why we chose to use HBase as a database, and predominantly using MapReduce as the prime method of calculating and processing data.

4.1 Storage Choice

To build a database for Wikipedia page view statistics the database must be scalable. The reason for this is the huge amount of data the page view statistics produce. Each day every hour Wikipedia releases the page view statistics for all HTTP requests Wikipedia receive. Every released file is about 300 Megabytes, amounting to around 7 Gigabytes each day. For a month this increases to 210 Gigabytes, and a year - 2.53 Terabytes. There are seven years of released page view statistics, and the amount of data is growing by the day. If one intends to use this vast amount of data, it is obvious that key features of the database need to be quick look-up times and scalability. In recent years, a lot of progress has been made within subjects such as cloud computing and big data. The databases used in these fields are typically NoSQL databases [25], and it makes perfect sense for us to take the same approach. A key feature of NoSQL databases is scalability, and many systems provide quick retrieval of records. As of today there exists many viable databases for this use, for example HBase, Cassandra, Google BigTable, Redis, Hazelcast and Distributed MySQL.

As we have some experience working with Clouderas Hadoop distribution, which contains HBase, it was fairly logical to choose HBase. HBase is also a part of the Apache Hadoop project, and is highly dependant on Hadoop, as HBase runs atop the Hadoop Distributed File system. The data we will work with in this thesis is also highly suited to the MapReduce programming paradigm, so the combination of HBase and Hadoop is a perfect match for our system.

4.2 HBase

HBase is an Extensible Record Store, or Wide Column Store [3], heavily modelled on Google's BigTable [4]. HBase consists of one master server, which monitors the region servers in the cluster, and several region servers. The region servers are responsible for a region in the data input, in this case an example can be that one region server has the responsibility of data ranging from A to E. ZooKeeper is an application used by HBase to coordinate the cluster [12, 31]. The architecture of HBase is described in Figure 4.1.

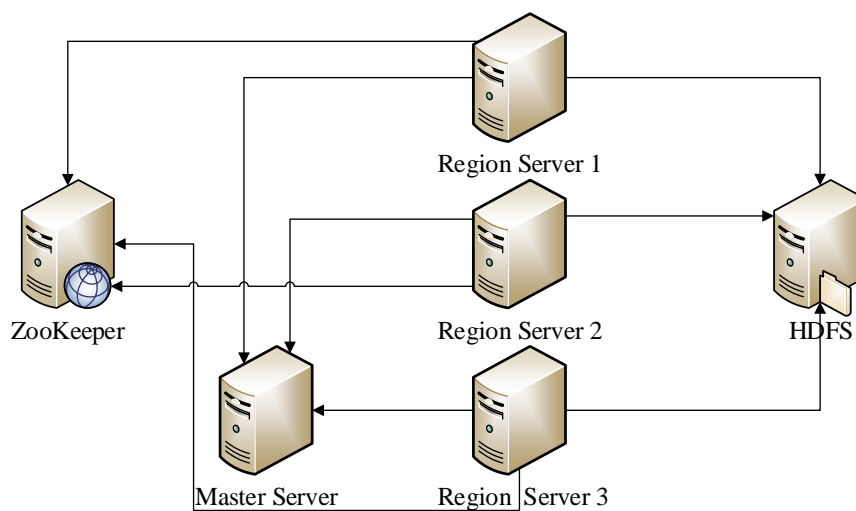


Figure 4.1: HBase Architecture

The table structure in HBase consist of rows, containing a RowKey and an amount of column families. A column family is a field which contains attributes with a desired value. When naming the column families, it is important to find a balance between how descriptive they are as to what the column family contain, and the length of the name. This is further elaborated in Section 4.2.3. Inside the column families you have qualifiers. For example in a column family 'Address', the qualifiers 'StreetName' 'StreetNumber' and 'ZipCode' could be included. The values themselves are placed inside the qualifiers, together with a timestamp. See Figure 4.2 for a graphical representation of HBase rows inside a table.

HBase does not store a value in each field for every single entry, it stores only the given values. If there is no value for a given field, it will be empty, thereby saving storage space. Similar to BigTable, HBase only stores arrays of bytes, and has no further concept of data types. As such, everything that can be converted into an array of bytes can be a value in an HBase table [4]. A column family can in some cases be looked at as a table in a relational database.

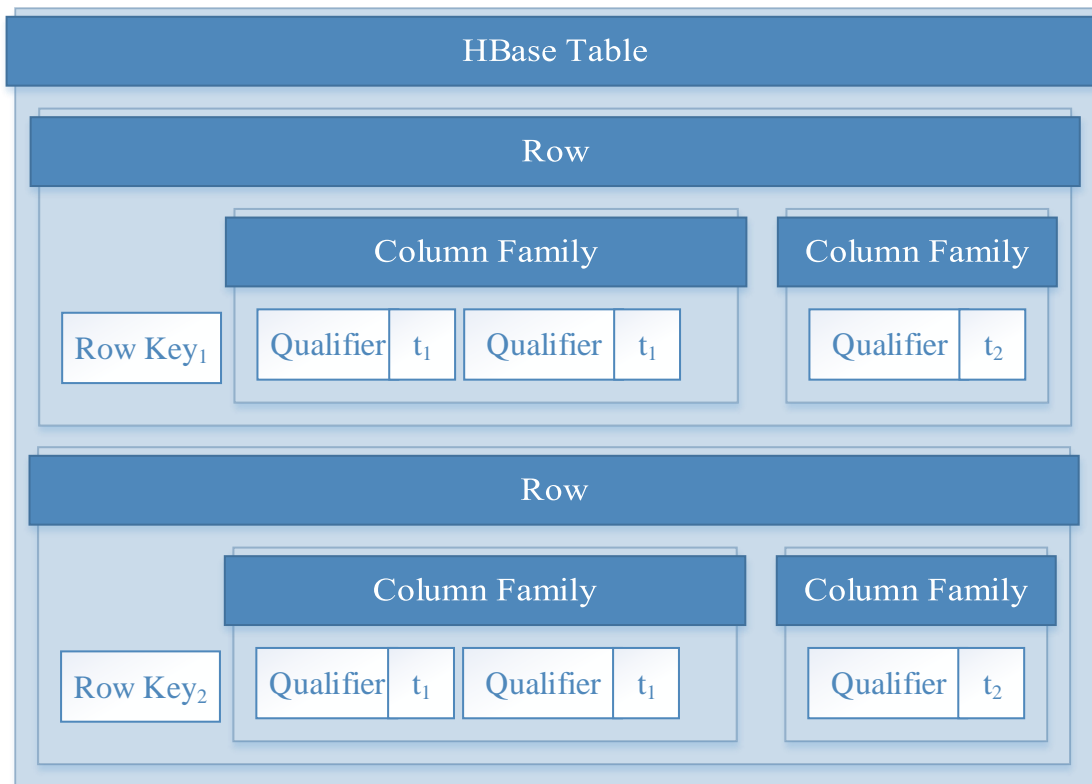


Figure 4.2: HBase Rows

4.2.1 Retrieving Data from HBase

There are numerous ways to retrieve data from HBase. It can be accessed through for example Java, Apache Hive and PHP. Since we want to build a web application, we choose to use PHP. This limits our alternatives to Apache Thrift, REST and JSON. Both an Apache Thrift and a REST server are implemented into the Cloudera Manager Free. Both these two servers are simple to use and it is easy to fetch the desired data, but our choice is to use Apache Thrift. For our use there are big differences between these two servers and therefore we had to pick one. See Figure 4.3 for an overview of the retrieval process.

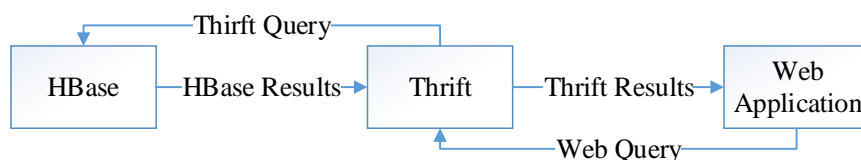


Figure 4.3: Communication - HBase - Thrift - Web Application

4.2.2 HBase RowKey Design

A well designed RowKey is crucial in order to achieve good performance from a HBase table. The RowKey is stored sequentially, sorted lexicographically on a byte for byte basis - see Table 4.1. This makes it act as an index on its own, and the design of the key then becomes perhaps the single most important design choice when creating a new table. With knowledge of the types of access patterns one will have towards the database, one can design the key such that all related rows are stored on the same region server [4, 12].

HBase RowKey Sorting		
RowKey	Content	Timestamp
person_11	address:city:oslo	t_1
person_111	address:city:trondheim	t_2
person_12	address:city:stavanger	t_3
person_abc	address:city:bergen	t_4

Table 4.1: HBase - Sorting of RowKeys

4.2.3 Disk Usage

According to [26], HBase uses a lot of disk space, more than similar data stores such as Cassandra, Distributed MySQL and Project Voldemort. The reason for this, they claim, is the additional schema, as well as the version information that is stored with the key-value pairs. There are multiple steps that can be taken in order to limit the storage space HBase uses. One is finding a balance between RowKey and column family size and usability. While RowKeys are useful for indexing the rows, for every entry, the full RowKey, column family names and qualifier names are stored. This means that limiting the size of column family names and qualifiers help reducing storage space used [12]. This has implications in the report, as we generally describe column families and qualifiers with more descriptive names than they actually have in the system - for increased readability. It is also possible to change the number of versions HBase stores in each column family - the default is 3. While we do some simple things in order to keep disk usage down in this project, the general approach is that we, design wise, disregard disk usage in favour of speed.

4.3 MapReduce

MapReduce is a programming model, mainly used for processing very large data sets, first introduced by Google [7]. In this project we utilise Apache Hadoop - the most popular open source implementation of MapReduce - through Clouderas distribution of the software. The programming model is based on two phases, phase one, which uses a Map function, and phase two, which uses a Reduce function. The basic input and output of these functions are pairs of keys and values. For graphical representation of the MapReduce paradigm, see Figure 4.4.

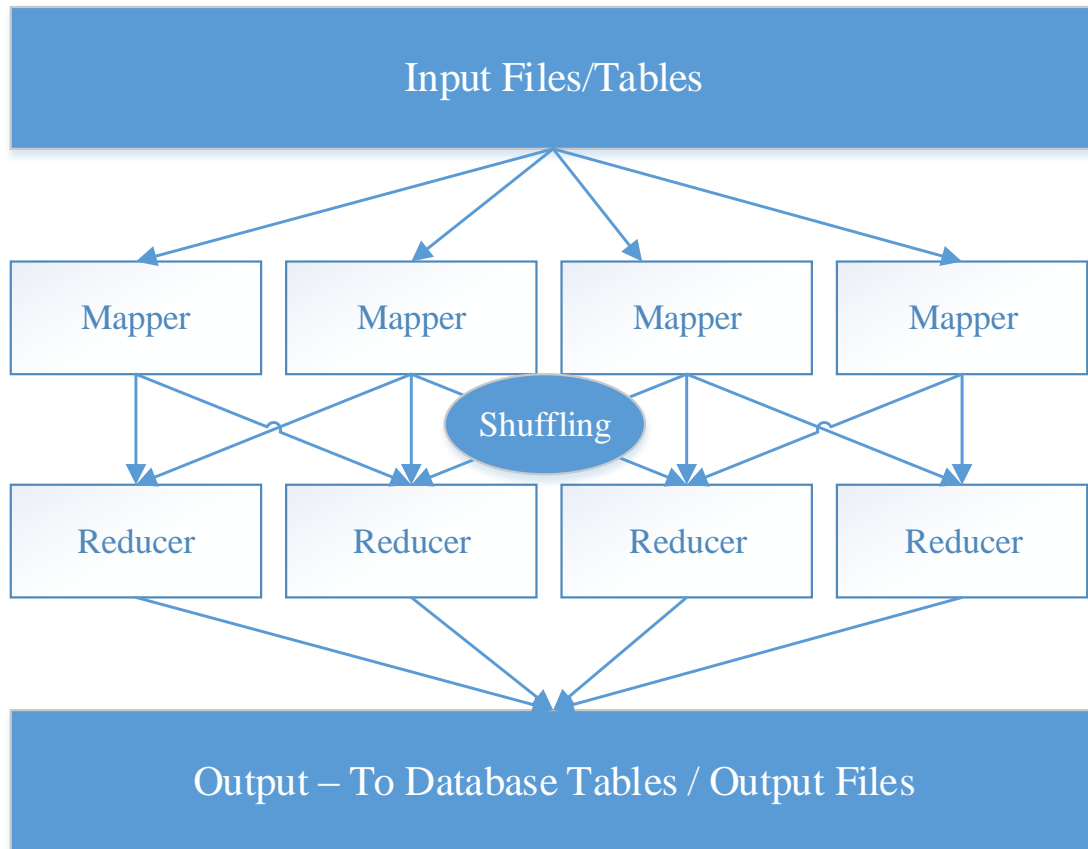


Figure 4.4: MapReduce Programming Paradigm

The first phase is the Map function. This function consists of processing the input - a list of keys and values $\langle K_1, V_1 \rangle$ - and create a new list of key value pairs $\langle K_2, List V_2 \rangle$. This is the Map functions output. For example, the input can be text, in which each lines offset from the beginning of the document is the key, and its content is the value. The output from the mappers then goes through a shuffling process, in which the key values that are hashed to the same value are sent to the same reducer.

The Reduce function processes its input $\langle K_2, List V_2 \rangle$ and merges the key values where they match and generates a new value for them. The output that the reducer produces is $\langle K_3, V_3 \rangle$. After all the output is generated, it is written to, and replicated in, the distributed file system [7, 20, 31].

4.3.1 Hadoop and HBase

As mentioned in Section 4.1, HBase is highly dependant on Hadoop, and HBase is even a part of Apaches Hadoop project, and is the preferred database for Hadoop¹. As a result of their close

¹<http://hbase.apache.org/>

relationship, there exist multiple functions tailored to altering the standard flow of MapReduce jobs to better suit communicating with HBase. Examples of this include taking entire HBase rows as input in the Map function, instead of lines in a file, and having database *Puts* directly to HBase as output, rather than generic Key-Values. It is also possible to altogether skip the Reduce step, and write directly to HBase in the Mapper - if there is no need for further processing of the data. However, certain functionalities do not yet exist, such as having multiple HBase tables as input [19]. Hadoop and HBase are open-source projects, and enhancements are added regularly.

4.4 DBpedia

DBpedia is a community driven effort to retrieve structured information from Wikipedia, systemize it, and make it publicly available on the internet [1]. The English version of the DBpedia knowledge base describes roughly 3.77 million entities². Out of these 3.77 million entities, about 2.35 millions are classified in a consistent *Ontology*, which include areas such as cities, diseases, bands, and other similar areas related to for example geography, (groups of) people, and works of art. The relations in DBpedia are presented in the form of triples, where the relation first present the subject, predicate and then the object [15]. An example of a triple like this can be:

Cesc_Fàbregas, currentclub, FC_Barcelona

The provided DBpedia files we utilize were created, according to DBpedia, using a Wikipedia dump from late May / early June 2012³.

4.4.1 Related Entities

There exist several opportunities to connect related entities from Wikipedia to each other. It is possible to utilize the link graph structure, as in [6]. Another solution is to use a DBpedia dump containing the linked entities from the Wikipedia info boxes for the entities, see the markings in Figure 4.5.

It is also possible to use a MySQL database, provided by Wikipedia⁴, containing all entity to entity relations. We experimented with the two last mentioned alternatives, before deciding what to choose. We found that the importing of the MySQL database was too time and space consuming compared to the DBpedia alternative. To mention some results from this experiment, the import of the MySQL database ran for around 24 hours and was still not completed, while the DBpedia file import took around 10 minutes. The Wikipedia MySQL database is however more up to date compared to the DBpedia alternative.

²<http://dbpedia.org/About>

³<http://wiki.dbpedia.org/Downloads38#h206-2>, accessed April 2013

⁴http://en.wikipedia.org/wiki/Wikipedia:Database_download

Cesc Fàbregas			
			
Fàbregas playing for Spain at the UEFA Euro 2012			
Personal information			
Full name	Francesc Fàbregas Soler ^[1]		
Date of birth	4 May 1987 (age 25) ^[1]		
Place of birth	Arenys de Mar, Spain		
Height	1.79 m (5 ft 10 1/2 in) ^[2]		
Playing position	Midfielder		
Club information			
Current club	Barcelona		
Number	4		
Youth career			
1995–1997	Mataró		
1997–2003	Barcelona		
2003	Arsenal		
Senior career*			
Years	Team	Apps [†]	(Gls) [†]
2003–2011	Arsenal	212	(35)
2011–	Barcelona	56	(19)
National team [‡]			
2002–2003	Spain U16	8	(0)
2003–2004	Spain U17	14	(7)
2005	Spain U20	5	(0)
2004–2005	Spain U21	12	(8)
2006–	Spain	78	(12)
2004–	Catalonia	2	(0)

Figure 4.5: Info Box Example - Cesc Fàbregas

4.4.2 Redirects

To have the possibility to implement this feature, we need a file containing redirects in Wikipedia. DBpedia provides such a file, although it is a bit outdated. The main usage of redirects in this system is getting to the right pages when an entity consists of any special sign, for example Nürburgring and Arsène Wenger. There exist other methods to do this, for example downloading a program to extract redirects through Wikipedia dumps, which in this case is too time consuming, and beyond the main focus point of the task.

Chapter 5

HBase Table Design

The database schema used in this project consists of seven tables. The 'Pagecount' table contains the information about any given entity on any given day. There are two burst tables containing occurrences of bursting entities, the 'EventDate' table, which is sorted on date, and the 'EventEntity' table, which is sorted on entity. Two tables are used in the calculation of Jaccard Similarities. In addition there is one table for redirects, and one table for relations. In this chapter, we describe these tables.

5.1 Pagecount Table

The RowKey in the 'Pagecount' table consist of the name of an entity and a date field:

```
'entity_YYYYMMDD'
```

The table has two column families, 'daily' and 'statistics' (in the system they are shortened to 'd' and 's' respectively, in order to conserve disk space). The 'statistics' column holds the total values for an entity, such as the accumulated values of page views and page edits, and the total number of page count files that have been processed. The other column family, 'daily', contains information about the total number of page views for a single entity on a single day, as well as the total amount of edits performed on that entity's article for the same day, in addition to values required for the calculation of bursts - see Chapter 6. The content in the 'statistics' column is only required to be stored in one row per entity. This row is marked 'entity_00000000'. Due to the sorting in HBase, all rows concerning this entity come directly after this row, making for rapid lookups - which is the most important feature in the design of this RowKey. This table also stores an intermediate value for each entity, used to calculate the standard deviation for all entries of an entity. See Tables 5.1 and 5.2 for an overview of the Pagecount table. It is worth mentioning that the 'daily' column family is empty for all rows on the form 'entity_00000000', and likewise, the 'statistics' column family is empty for all rows on the form 'entity_YYYYMMDD'. The 'FilesScanned' qualifier is 62 because we have two data points from each day for a complete month.

The total number of files in the 'statistics' column family is a number that for all accounts and purposes will be similar for all entities of interest, and saving it for every entity is sub-optimal.

However, this number will be below 10 bytes for the foreseeable future, and there are currently around 4,2 million articles on the English Wikipedia¹, and the amount of storage space this takes up is negligible compared to the the other content we are storing. It is stated in the HBase documentation that one should be weary with different column families having lopsided number of entries², but as it in our case is only talk about thousands of records per entity, we anticipate no issues regarding this. The alternative is to split the table into two tables. However, the HBase version we are using does not yet straight forwardly support having multiple HBase tables as input in a MapReduce Job³, something that has been mentioned as a problem in other projects [19]. However, having multiple tables as input in a MapReduce job is scheduled for an upcoming version of HBase⁴ so this is an issue that hopefully will be trivial to improve in the future.

Pagecount Table				
RowKey	Column Family			
Entity_date	Statistics			
	'TotalHits'	'TotalEdits'	'FilesScanned'	AbsoluteDeviation'
bryan ruiz_00000000	7575	117	62	754
cesc fàbregas_00000000	35612	213	62	1095
mikel arteta_00000000	4153	257	62	327
romelu lukaku_00000000	22497	150	62	742

Table 5.1: HBase - Pagecount Table Part 1 - Statistics Column Family

Pagecount Table				
RowKey	Column Family			
Entity_date	Daily			
	'Z-score'	'Modified Z-score'	'DailyHits'	'DailyEdits'
bryan ruiz_20110831	111	71	4311	51
cesc fàbregas_20110815	4	24	5323	39
mikel arteta_20110831	32	60	1907	251
romelu lukaku_20110806 ⁵	0	0	3868	22

Table 5.2: HBase - Pagecount Table Part 2 - Daily Column Family

5.2 Burst and Event Tables

The system could have been realised by only using one table for bursts and events, but having multiple tables with differently designed RowKeys secures fast query response times for

¹https://en.wikipedia.org/wiki/English_Wikipedia, accessed the 19th of April 2013

²<http://hbase.apache.org/book/number.of.cfs.html>

³<http://www.cloudera.com/content/cloudera/en/products/cloudera-enterprise-free.html>, as of May 7th 2013, uses HBase version 0.94.2

⁴<https://issues.apache.org/jira/browse/HBASE-3996>

⁵This entry would most likely result in a burst, but as it is in the very beginning of the data set, the window is not yet full, and no calculation of Z-scores/Modified Z-scores is performed

all parts of the system. Thus, we sacrifice disk usage for speed, as elaborated in 4.2.3. This approach is also evident in that much of the information contained in these tables is information that is already being stored in the 'Pagecount' table, but we prefer tables that are tailored for their specific tasks, and not for efficient space consumption. Two column families are present in each of the two tables, one with events, and one with bursts. Both these denote a spike in page views, but in order to be labelled as an event, there needs to be a higher amount of edits than some threshold - as elaborated in Chapter 3. An 'Event' will not be stored in the 'Burst' column family, and vice versa - depending on whether it is a Burst or an Event that is stored. This means that there is always one empty column family in a row.

5.2.1 EventDate Table

The purpose behind the 'EventDate' table is to quickly access all bursts and events on a given day, as well as to give a certain indication of the magnitude of the burst or event. In order to accomplish this, the RowKey is designed the following way:

YYYYMMDD_X_entity

The 'X' in the key is a single numerical digit, where a higher number indicates a greater burst. This gives the end user some control when deciding what to consider as a relevant burst/event. As the timestamp is placed first in the RowKey, this acts as a natural index, meaning that all bursts or events on a single day are placed and can be retrieved sequentially. There are two column families. One is written to in the case of an event, in other words, a burst that has a sufficient number of edits associated with it. The qualifiers in the 'Event' column family are 'PageHits', 'PageEdits' and 'Z-score/Modified Z-score'. The other column family is the one that is written to if there is a burst, but not enough edits for it to be considered as an event. It stores the same information, barring 'PageEdits'. The 'Z-score' qualifier in the Event and Burst column families is either the regular Z-score or the Modified Z-score - depending on the method is used. For an overview, see Tables 5.3 and 5.4.

EventDate Table			
RowKey	Column Family		
Date_burstlevel_Entity	Event		
	'PageHits'	'PageEdits'	'Z-score'
20110829_2_davide santon	999	31	23
20110831_3_bryan ruiz	4351	51	111
20110831_3_mikel arteta	1907	251	32

Table 5.3: HBase - EventDate Table Part 1 - Column Family Event

5.2.2 EventEntity Table

The main purpose of this table is to have easy access to all the bursts and events for an entity, something the previously mentioned 'EventDate' table does not manage well. The column

EventDate Table		
RowKey	Column Family	
Date_burstlevel_Entity	Burst	
	'PageHits'	'Z-score'
20110815_2_kolo touré	542	29
20110815_2_mikel jon obi	351	23
20110822_1_tom cleverly	2412	12

Table 5.4: HBase - EventDate Table Part 2 - Column Family Burst

family and qualifiers of this table is structured in the same way as the 'EventDate' table. For an overview, see Tables 5.5 and 5.6. The big difference is how the RowKey is designed. It changes the sequence of the entity and the timestamp, and it does not have any indicator of the magnitude of a burst or an event:

entity_YYYYMMDD

With the entity being the first part of the RowKey, retrieving all entries regarding the same entity will be fast. There is no indicator of the magnitude of the burst or event as in the previous table, as the number of bursts or events for a single entity will be so low that one quickly can get an overview over all of them. As in the 'EventDate' table, the final qualifier in the Event and Burst column families, 'Z-score' is either the regular or Modified Z-score - depending on the method is used.

EventEntity Table			
RowKey	Column Family		
Entity_Date	Event		
	'PageHits'	'PageEdits'	'Z-score'
bryan ruiz_20110831	4351	51	111
davide santon_20110830	999	31	23
mikel arteta_20110831	1907	251	32

Table 5.5: HBase - EventEntity Table Part 1 - Column Family Event

EventEntity Table		
RowKey	Column Family	
Entity_Date	Burst	
	'PageHits'	'Z-score'
kolo touré_20110815	542	29
mikel john obi_20110815	351	23
tom cleverly_20110822	2412	12

Table 5.6: HBase - EventEntity Table Part 2 - Column Family Burst

5.3 Redirect and Related Entities Tables

The 'Redirect' and 'Related Entities' tables are quite similar, therefore they are described in the same section. They both consist of a RowKey and one column family containing one qualifier. In the 'Related Entities' the content of the column is one related entity. If the entity has more than one related entity it is stored in a new row. This could have been realised by putting all the related entities into one row, but this would have demanded much more processing in the front end of the system, and therefore we choose to put each related entity in a separate row. We solve this by adding an underscore and a number to each RowKey, to be able to quickly scan all related entities for an entity. See Table 5.7.

Related Entities Table	
RowKey	Column Family
Entity_ <i>x</i>	RelatedTo
	'Related Entity'
cesc fàbregas_1	arsenal f.c.
cesc fàbregas_2	fc barcelona
cesc fàbregas_3	spain

Table 5.7: HBase - Related Entities Table

In the Redirect table, we use the redirect itself as the RowKey. The contents of the row is the actual entity the redirect leads to. By designing the table in this way, it is very fast to query the table in order to see if a search string has a possible redirect. We do not need to append any digits to the RowKey, as each redirect is unique. The underscore is appended in order to ease retrieval in the front end. See Table 5.8:

Redirect Table	
RowKey	Column Family
Entity_	RedirectsTo
	'Redirect Entity'
cesc fabregas_	cesc fàbregas
francesc fabregas soler_	cesc fàbregas

Table 5.8: HBase - Redirect Table

5.4 Jaccard Tables

The examples for these tables use the Modified Z-score. The first table, 'JaccardString' - Table 5.9, contains the date of the first burst for each entity and a string of zeroes and ones, where a zero marks no bursts at a date, and a one marks a burst. This string is used to calculate the score for entities that is placed into the 'JaccardScore' table. The RowKey in both these tables consists of an entity name, but only in the 'JaccardScore' table do we add an underscore and a

number after the entity, because this is the only table that is used by the front end.

JaccardString Table		
RowKey	Column Family	
Entity	JaccardStatistics	
	'FirstBurst'	'BitString'
davide santon	20110829	0000000000000000000111
steve jobs	20110826	0000000000000000111100
tim cook	20110826	0000000000000000111100

Table 5.9: HBase - JaccardString Table

The 'JaccardScore' table, see Table 5.10, is similar to the 'Related Entities' table, Table 5.7. The columns are filled with entities that has a Jaccard similarity with the RowKey. This table could have been created by filling columns with all the Jaccard similar entities to the RowKey, but this would not be a scalable solution, and would slow down the front end significantly due to a lot of processing. As all our other designs, this design favours favour speed and not space consumption. The digits added at the end of the RowKey may be the same, and this is due to the fact that they are added at the same time, to decrease the run-time of the system.

JaccardScore Table	
RowKey	Column Family
Entity_ <i>x</i>	JaccardStatistics
	'Entity Score'
mike flanagan (baseball)_1	steve jobs_1.0
mike flanagan (baseball)_2	tim cook_1.0
steve jobs_1	mike flanagan (baseball)_1.0
steve jobs_3	tim cook_1.0
tim cook_2	mike flanagan (baseball)_1.0
tim cook_3	steve jobs_1.0

Table 5.10: HBase - JaccardScore Table

Chapter 6

Burst and Event Detection

In this chapter we describe our approach to burst and event detection, and go into detail about how we have implemented this in the system.

6.1 Sliding Window Implementation

The first task in regards to detecting outliers with the Wikipedia page view files as source, is to identify the days where the number of views is abnormally higher than it usually is. We achieve this by applying a sliding window-approach, as in [21]. We apply two different means of outlier detection. These two methods will be compared in Chapter 10.

Naturally, the following algorithms look very much the same, as it is in fact the same code snippet. The actual detection and distinguishing between bursts and events are further elaborated in Section 6.2. First, the purpose of Algorithm 1 is to for each entity, for each day, find the standard deviation and mean page views of the n last days, where n is the size of the window. We then calculate the number of standard deviations - or Z-score - the value in question deviates from its window. For more information about the mathematics and statistics behind this, see section 3.3. Algorithm 2 shows the same algorithm for calculating the modified Z-scores. The purpose here is to for each entity, for each day, find the median page view count of the last n days. In the next step, for all of the n last days, we subtract the median from every element. After doing this, we again find the median of this new list - the Median Absolute Deviation. With these values, we calculate the Modified Z-score - elaborated in Section 3.3.2.

Algorithm 1 Sliding Window Implementation - Z-scores

```

1: reduce(key, values)
2: TreeMap<String, Integer> dateValueMap;
3: for elements in values do
4:   Parse Entity and DailyHits from values
5:   Add Entity and DailyHits to dateValueMap
6: Create list from DailyHits in dateValueMap
7: for i = 0 → size of list do
8:   if window not full then
9:     window[i] = list[i]
10:  else
11:    for j = 0 → length of window do
12:      tempSum += window[j]
13:    windowAvg = tempSum / WINDOWSIZE
14:    for i = 0 → length of window do
15:      windowDev += (window[j] - tempAvg)2
16:    standardDeviation = sqrt(windowDev / WINDOWSIZE)
17:    zScore = (DailyHits - windowAvg) / standardDeviation
18:    window[i mod WINDOWSIZE] = list[i]
19:  Write to HBase

```

As indicated in line 1, everything happens inside MapReduce's reduce function, and everything is performed for all keys. The key is the name of the entity. The entries in the *values* list are on the form *YYYYMMDD_count*. Lines 3 through 5 deal with adding the contents of the *values* list into a data structure that is easier to work with. A *TreeMap* is chosen, due to the inherent sorting on its keys. Some manipulation is done on the key and value in this *TreeMap*, to ensure that the entries in the map are in the desired sequence. Line 19 creates a *list* based on the values in the aforementioned *TreeMap*. This means that the values in this list are the *DailyHits* for an entity, in chronological sequence. Lines 8 and 9 fill the *window* list with values. No calculations are performed until the *window* is full. As a result of this, the first few entries in the dataset will be unable to burst. The For loop at Line 11 sums the values in the window. The For loop at line 14 calculates an intermediate value in regards to finding the windows standard deviation, before the actual standard deviation is calculated in line 16. In line 17 we calculate the Z-score for the value, which is the one that is written to HBase. In line 18, the window is updated by replacing the oldest value with the current one.

For the calculation of our modified Z-scores, see the following Algorithm 2. This algorithm is not very different from Algorithm 1, in fact the first seven lines are identical. Further, we use two lists instead of one in this algorithm, *window* and *unmodifiedWindow*. The *unmodifiedWindow* keeps track of the actual unmodified values in the window, while the *window* is a copy of *unmodifiedWindow*, and used for calculating the median and MAD of the values. Lines 8 and 9 fill the window, and lines 11 and 12 copies the content of *unmodifiedWindow* into *window*. Lines 13 to 18 perform the aforementioned calculation of medians and MADs. in line 16, we use the absolute value, as we do not want negative values in the list. Lastly, we calculate the Modified Z-score and write this to HBase.

Algorithm 2 Sliding Window Implementation - Modified Z-scores

```

1: reduce(key, values)
2: TreeMap<String, Integer> dateValueMap;
3: for elements in values do
4:   Parse Entity and DailyHits from values
5:   Add Entity and DailyHits to dateValueMap
6: Create list from DailyHits in dateValueMap
7: for i = 0 → size of list do
8:   if unmodifiedWindow not full then
9:     unmodifiedWindow[i] = list[i]
10:  else
11:    for j = 0 → size of window do
12:      window[j] = unmodifiedWindow[i]
13:    sort window
14:    median = median of window
15:    for j = 0 → size of window do
16:      subtract median from window[j]
17:    sort window
18:    MAD = median of window
19:    Modified Z-score = 0.6745 * (DailyHits - median) / MAD
20:    unmodifiedWindow[i mod WINDOWSIZE] = list[i]
21:  Write to HBase

```

HBase Row				
RowKey	Values			
	'Z-score'	'Modified Z-score'	'DailyHits'	'DailyEdits'
cesc fàbregas_20110815	4	24	5323	39

Table 6.1: Example Row, as Generated by Algorithms 1 and 2 and Used by Algorithm 3

6.2 Burst and Event Detection

As we now have the Z-score or Modified Z-score respectively, for all entities for all days, the task that remains is to compare each separate score against the thresholds we have chosen. This is done in a MapReduce map function, that for each row in an HBase table, retrieves the Z-score/Modified Z-score, daily page views (DailyHits) and number of edits (DailyEdits) (see Table 6.1). Line 3 - 5 in Algorithm 3 checks if the values exceed the different thresholds we have set. All these thresholds are further elaborated in Section 6.2.1.

Algorithm 3 Event and Burst Detection

```
1: map(HBase row, values)
2: Get DailyHits, DailyEdits, Z-score/Modified Z-score from HBase Row
3: if DailyHits > dailyThreshold then
4:   if score > scoreThreshold then
5:     if DailyEdits > editThreshold then
6:       Create Event
7:     else
8:       Create Burst
9:     Write to HBase
```

The output from this algorithm is written to two different HBase tables, with different RowKeys. This is explained in Section 5.2.

6.2.1 Choosing Window Size and Thresholds

Some important decisions to make is what window size to use for Algorithms 1 and 2, as well as the thresholds to use in Algorithm 3.

Window Size

Zhao et al. [37] have a window size of 10 hours, that is 10 data points, when finding bursts on Twitter, and [21] use a window of 48 hours, that is 48 data points, when performing First Story Detection, also on Twitter. This is not a value that is transferable to our system, as we operate on a different granularity - days versus hours. 2 days is naturally too short to perform burst detection on a static data set, but it seems to do well on a volatile data set such as twitter. However, if translated to data points, the situation changes. We use two data points per day, and if we were to use 48 data points, the window would consist of 24 days, which may be a more realistic value. We have ended up with a window size of 10 days - 20 data points - for the regular Z-scores, and 11 days for the modified Z-scores. The one day difference is in regard to the simplicity of the implementation. The most important aspect when choosing window size is that it is big enough to enfold both the bursty patterns, as well as the status quo. This is also not a value that is cast in stone, and can be changed by a rerun of the system on the data set.

Thresholds

First, we want a threshold that considers how large a daily page view count have to be in order to be considered as bursting. This is done to prevent insignificant entities from bursting all the time. The assumption is that if an entity generally have very few page views, it will have a low mean and low standard deviation. This again makes it easier for the entity to burst with the slightest increase in daily page views. This assumption is supported from early test results, where entities with general low amount of page views burst more often and without reason. To counter this we empirically set a threshold saying that the Daily Hits needs to be above 250 for

an entity to be considered bursting.

The second threshold decides if a Z-score or Modified Z-score is a burst or not. [9], [16] and [37] operate under the assumption that values in the area of 3 - 3.5 standard deviations/modified Z-score should be treated as outliers with similar tests as ours. Preliminary results from our system show that a threshold as low as this yields a very high number of bursts, with a lot of questionable results for both regular and modified Z-scores. We have instead decided to set the lowest value indicating a burst to 10 - with some additional levels at 20 and 30 in order to allow the user to focus on the greatest bursts. These are values that can be changed according the user preference by running the system with desired thresholds.

As for what threshold should be used to distinguish events from bursts in Algorithm 3, [21] propose 10 edits in Wikipedia required in the 12 last hours. We have taken a similar approach, with 20 edits over the course of a day enough to be treated as an event.

Chapter 7

Back End

In this chapter we explain in greater detail how we store the information from the Wikipedia page view files and edit statistics into HBase, and in general tie the whole back end of the system together. The process of populating the burst tables is described in detail in Chapter 6, and will not be described in this chapter. A general overview of the back end can be seen in Figure 7.1.

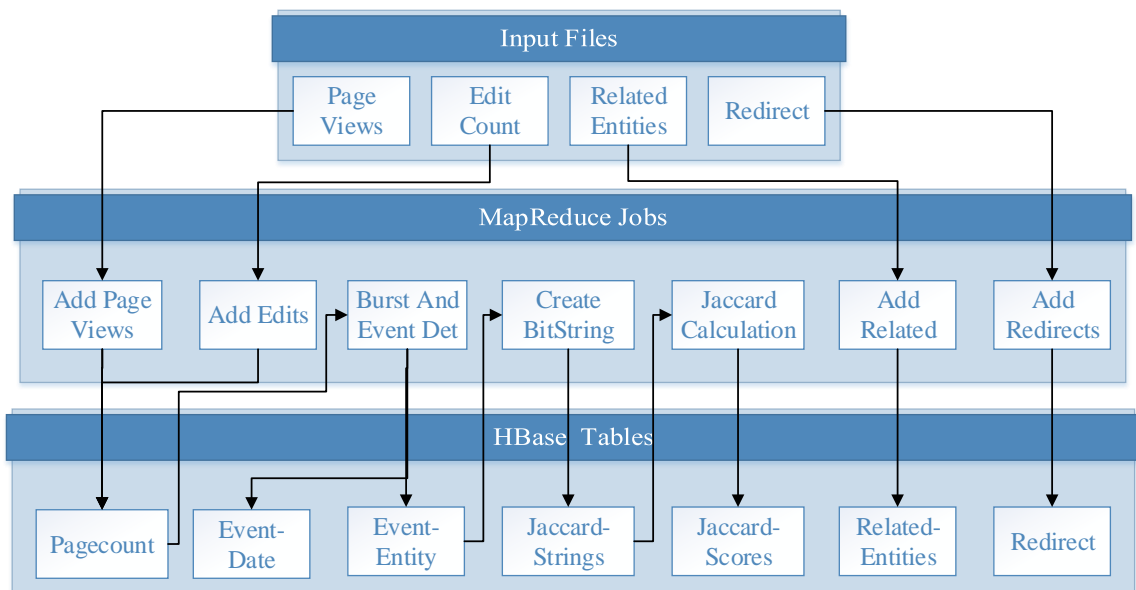


Figure 7.1: Overview of the Back End

7.1 Adding Page View Statistics

Adding statistics of each entity's daily page views is an essential part of the system. The system is able to generate this daily page view count from the raw page view statistics files supplied

by Wikipedia¹. These files are released on an hourly basis, and contain information about all HTTP-requests to any national Wikipedia or Wikipedia Projects page, including requests for non-existing pages. A sample line from these files follow:

en Davide_Santon 58 1102649

The line consist of four parts. The first denotes the version of Wikipedia - here, the regular English Wikipedia. The second, Davide_Santon is the page, or entity, being requested. The third is the actual number of requests - 58, and the fourth is the size of the content returned - 1102649. We only care about the first three elements. We are operating exclusively on the regular, English Wikipedia, meaning we are only interested in lines starting with "en ".

Algorithm 4 Adding Daily Page View and Total Statistics

```

1: map(key, value)
2: Extract line from value
3: if line starts with "en " then
4:   split line
5:   Entity = line[1]
6:   filter Entity
7:   decode Entity
8:   countAndTimestamp = line[2] + timestamp
9:   send (Entity, countAndTimestamp) to reducer
10:
11: reduce(Entity, values of countAndTimestamps)
12: TreeMap<String, Integer> dateValueMap;
13: for countAndTimestamps in values do
14:   Parse Timestamp and Page View Count
15:   Calculate DailyHits
16:   Add Entity and DailyHits to dateValueMap
17: Create list from DailyHits in dateValueMap
18: for i = 0 → size of list do
19:   Write DailyHits to HBase
20: Write Statistics to HBase

```

As mentioned in 6.1, this part of code in our system performs multiple tasks, so the algorithms are overlapping. Here we include the map function, which is largely irrelevant for aforementioned algorithms. The 'filter entity' step in line 5 excludes HTTP requests to non-standard Wikipedia pages, such as user pages and direct link to images. Line 7, 'decode entity', is elaborated in the next subsection. The rather odd notion of merging the page view statistic and timestamp in a single String is done in order to do all calculation of Sliding Windows and daily page view statistics in a single MapReduce job. The timestamp is retrieved from the name of the current input file, which contains this information. In line 20, we write to the 'statistics' column family, but we have omitted the details of the calculation of the values we write, as they

¹<http://dumps.wikimedia.org/other/pagecounts-raw/>

are trivial to calculate. For more details regarding the reduce function, see Section 6.1.

Decoding from Wikipedia to HBase

As mentioned in 4.2, HBase only saves arrays of bytes. The Wikipedia page view count files only contain regular letters and numbers. All special characters (for example, the Norwegian letters æ, ø, å) are percent-encoded. That means that each special character has a special code, starting with a percent sign (%). We decode this to regular UTF-8 before we add data to HBase. Doing this also prevents us from having to decode and process the results when querying from the web application. In addition, it increases readability of the entities when we are testing our own system, and it gives a certain level of consistency across our system. Decoding from the Wikipedia page view files is done through the native `java.net.URLDecoder.decode()` function.

7.2 Edit Count

The system uses a pre-processed file to add the edit count history into the database. This file was created out of a Wikipedia dump dated January 2nd, 2013. The file is structured using an entity to start each line, an identification number, followed by the date of each edit, as shown below.

$$\textit{entity someID date}_1, \textit{date}_2, \textit{date}_3, \dots, \textit{date}_n$$

The reason for using a pre-processed file is both the resources, especially disk space, available, and the time consumption that would have been needed to create this ourselves. A Wikipedia dump containing all the edit history for all entities is in the magnitude of multiple Terabytes, which is too large for us to handle. After pre-processing, the file is 11.3 Gigabytes.

To add the edit count statistics, the system uses the pre-processed file. The system reads the file line by line, split each line and put all the dates into a list. Intuitively, the length of this list is the total number of edits for this entity. This number is added to the 'statistics' column family. How we extract the number of total edits for each entity per day is explained in lines 5 to 12 in Algorithm 5.

7.3 Calculating Jaccard Coefficient

The calculation of the Jaccard coefficient is an operation that is carried out in two separate jobs. The first job consists of scanning the 'EventEntity' table to get hold of all bursts and event dates for every entity. Further the system creates a bit string where a '1' symbolise a burst and '0' symbolise a date where no burst occurs. The Strings are only written to the 'JaccardStrings' table if the entity has experienced more than one burst. The entities that have experienced one burst or less are very plentiful, and do not produce interesting results. The second job reads all entries from the 'JaccardStrings' table. Each entity is then compared against the others using

Algorithm 5 Adding Edits to Database

```

1: map(line)
2: Extract Entity from line
3: Populate dateTable with editDates
4: TotalEdits = length of dateTable
5: Write TotalEdits to statistics column family
6: dateValueMap<editDate, Count>
7: for i = 0 → length of dateTable do
8:     Increment count in dateValueMap
9: for editDates in dateValueMap do
10:     write editDate and Count to daily column family
11: Write to HBase

```

Equation 3.8. If the results, after deploying this formula, is above 0.8, we write to the 'Jaccard-Scores' table. Algorithm 6 describes how the 'JaccardScores' table gets populated.

Algorithm 6 Adding Jaccard Coefficients

```

1: Count
2: Extract bitStrings from JaccardStrings
3: Populate bitStringTable with bitStrings
4: for i = 0 → length of bitStringTable do
5:     for j = i + 1 → length of bitStringTable do
6:         Calculate Jaccard Coefficient for i and j
7:         if Coefficient > 0.8 then
8:             Write row i_count with column j_score
9:             Write row j_count with column i_score
10:        Increment count

```

To better understand line 8 and 9, see Table 5.10. It is also worth noting that this is not a MapReduce job, as doing this in MapReduce would be a more complicated affair.

7.4 Related Entities

As explained in Section 4.4.1, we choose to use a DBpedia provided file for the connection between entities. The related entities file retrieved from DBpedia² consists of triples looking like this:

```

<http://dbpedia.org/resource/Cesc_Fàbregas>
<http://dbpedia.org/ontology/team>
<http://dbpedia.org/resource/FC_Barcelona> .

```

²<http://wiki.dbpedia.org/Downloads38>


```
<http://dbpedia.org/resource/Cesc_Fàbregas>
<http://dbpedia.org/ontology/birthDate>
"1987-05-04"^^<http://www.w3.org/2001/XMLSchema#date> .
```

As seen from the examples, there exist different types of triples in the file. The triples needed by the system to relate entities to each other are the triples that contain a resource as the last part. In the example given above, it is only the first triple that is interesting for the system, as it links from and to a resource. Due to the different types of triples, we pre-process the file to reduce the amount of data that have to be processed. A program was created to pre-process the file. First it removes all triples that do not contain the prefixes given below.

```
http://dbpedia.org/resource/X
http://dbpedia.org/ontology/Y
http://dbpedia.org/resource/Z
```

Furthermore, the program remove all prefixes. Lastly, it writes the entity to a file where each entity have one line, each followed by the related entities. This pre-processing reduces the file size from 2.8 Gigabytes to 221 Megabytes. An example of how an entity looks in the file after pre-processing:

```
cesc_fàbregas fc_barcelona arsenal_f.c. spain
```

The adding of related entities is straightforward. The system use the pre-processed file as input and reads it line by line. Every line is split when a white space occurs, and all strings generated by this split are added to a list. We then iterate over this list, and add all elements to an HBase table where the entity name is the RowKey, and each related entity is placed as an entry in a column family - see Table 5.7.

7.5 Redirects

The approach concerning redirects is quite similar to the approach we use when adding related entities to the system. The DBpedia file³ is structured in the same way as the related entities file and is pre-processed to remove the prefixes. Two different examples of triples are shown below.

```
<http://dbpedia.org/resource/Cesc_Fabregas>
<http://dbpedia.org/ontology/wikiPageRedirects>
<http://dbpedia.org/resource/Cesc_Fàbregas> .
```

³<http://wiki.dbpedia.org/Downloads38>

```
<http://dbpedia.org/resource/Cesc_fabregas>  
<http://dbpedia.org/ontology/wikiPageRedirects>  
<http://dbpedia.org/resource/Cesc_Fàbregas> .
```

The examples above show that Wikipedia, in contrast to our system, is sensitive to capital letters. The fact that our system is not capital case sensitive makes the second example triple redundant, and it can be removed from the redirect list. Below we show which type of triples we treat as relevant.

```
http://dbpedia.org/resource/X  
http://dbpedia.org/ontology/wikiPageRedirects  
http://dbpedia.org/resource/Z .
```

The pre-processing on this file reduces the size from 846 Megabytes to 224 Megabytes. An example of how a redirect looks in the new file:

```
'cesc_fabregas cesc_fàbregas'  
'francesc_fabregas_soler cesc_fàbregas'
```

To integrate this feature into the system much of the same code used to implement the related entities is reused. The way this is implemented is to read each line and split lines on whitespace and write to HBase. This table is though structured a bit different, due to how the lookups are conducted. The RowKey in this table is the redirect, and not the real entity. With this design, a lookup in the table is done each time the database is queried for an entity - which is very quick due to the lookup being done on a database tailored for the task. If there is a match, the system suggests the entity contained in that row.

Chapter 8

Front End

The front end of this system is a prototype web service coded in PHP. The main purpose of the front end is to show functionality. The communication between HBase and the web service is conducted using an Apache Thrift server which fetches the HBase data and returns data to the web application. All queries in this system use a function, contained in Apache Thrift, called 'scanForPrefix'. When scanning HBase using 'scanForPrefix' we append an underscore after the query string, due to the way we have designed our RowKeys. This way, we know that the system searches for a whole entity or date. In Figure 8.1 we display the different HBase tables queried in the various views in the front end.

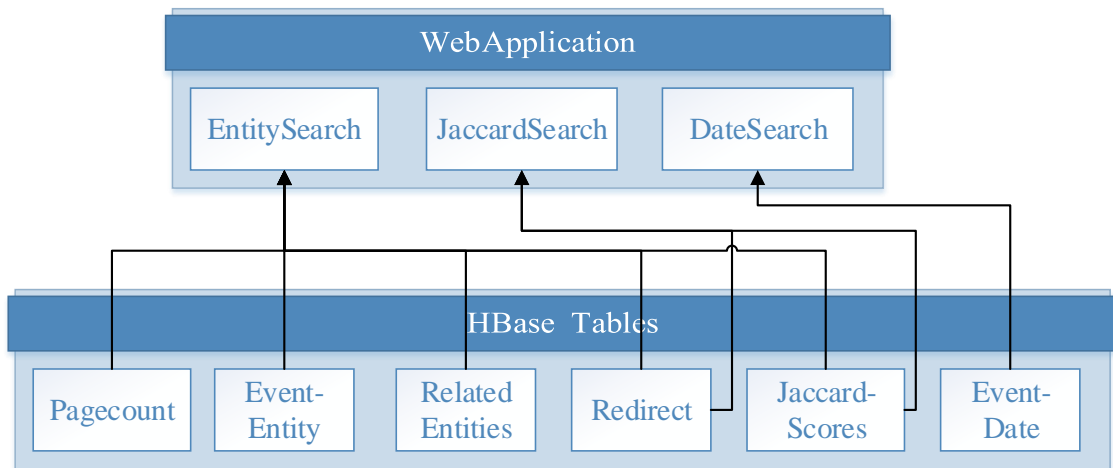


Figure 8.1: Overview of the Front End

The RowKeys fetched from HBase are encoded from UTF-8 to a format that does not contain letters with special signs. This is handled in PHP by the usage of a method called `mb_convert_encoding`, which converts the string from a desired encoding to another encoding, which in this case is from 'UTF-8' to 'HTML-ENTITIES'. All integer values derived from HBase tables are written as a byte array with four values. To calculate the real value of these arrays, the following formulas, Equation 8.1 and 8.2, are used. In these, x is the value of an integer.

$$array = (v_4, v_3, v_2, v_1) \quad (8.1)$$

$$x = v_1 + (v_2 \times 2^8) + (v_3 \times 2^{16}) + (v_4 \times 2^{24}) \quad (8.2)$$

The front end is divided into three views, Section 8.1 - 'Search for Event/Burst'. Section 8.2 - 'Search for Entity' and Section 8.3 - 'Jaccard Similarity Search'.

8.1 Search for Events and Bursts

In this view, see Figure 8.2, the user can select a date from a calendar and search for bursts or events, by selecting burst or event through radio buttons. Selecting nothing returns both. This queries the 'EventDate' table. To fetch the bursts/events, the web interface uses the query described earlier - 'scanForPrefix'. For a burst it queries for the 'burst' column family, and for events it queries the 'event' column family, see Table 5.3 and 5.4. In addition the user has the opportunity to select a burst level, which only displays the entities that is contained in the desired burst level. This is realized by editing the query string which is passed on to HBase. A normal query string when no burst level buttons are selected looks like this: 'YYYYMMDD_', and fetches bursts of all levels. When a burst level button is selected, the query string gets modified to 'YYYYMMDD_*x*_', where *x* denotes the desired burst level.

In the result table the user have two choices, either to go to the 'Search for Entity' view, to show the graphs and statistics, or Google the entity with the date appended, as in [6]. The Google search is conducted on the English version of Google in order to get the best possible results from these queries.

Search for Burst or/and Event

[Burst/Event Search](#) - [Entity Search](#) - [Jaccard Search](#)

22 August 2011 Event
 Burst

Burst level

- Level 1
 Level 2
 Level 3

Search for bursts

Searched For 2011-08-27 and Burst Level 3
 Total events: 12

Entity Name	Z-score	Page edits	Page hits	Burst Level	Go to graph	Google it
Andrea Poli	64	39	297	3	Andrea Poli	Google
Atiyah Abd Al-rahman	1899	42	2818	3	Atiyah Abd Al-rahman	Google
Franco Di Santo	38	39	263	3	Franco Di Santo	Google
Hurricane Irene	85	473	1662	3	Hurricane Irene	Google
Javaris Crittenton	1324	37	1968	3	Javaris Crittenton	Google
Let's Kill Hitler	291	28	3519	3	Let's Kill Hitler	Google
Mauritius	324	36	12432	3	Mauritius	Google
Nintendo World Championships	419	26	1256	3	Nintendo World Championships	Google
Park Chu-young	147	165	2440	3	Park Chu-young	Google
Saffir-simpson Hurricane Scale	35	21	5812	3	Saffir-simpson Hurricane Scale	Google
Taufic Guarch	58	22	268	3	Taufic Guarch	Google
Ulises Dávila	162	26	2192	3	Ulises Dávila	Google

Figure 8.2: Search for Burst/Entity - Example View - 31. August 2011

8.2 Search for Entity

In the 'Search for Entity' view, the user can enter a text string to query HBase. This will return all related data for this specific text string that is contained in HBase. The data the query returns then gets translated as described earlier. This view queries five of the seven HBase tables at least once, to get a hold of all the desired data that needs to be displayed. In Figure 8.3 we show an example view. In the following subsections every part of this view will be explained in detail, both textual and with the help of figures. At the bottom of this view we have implemented a button that is visible if there exist any entities that has a high Jaccard similarity to the searched entity. This button sends the user and the entity to the 'Jaccard Similarity Search'.

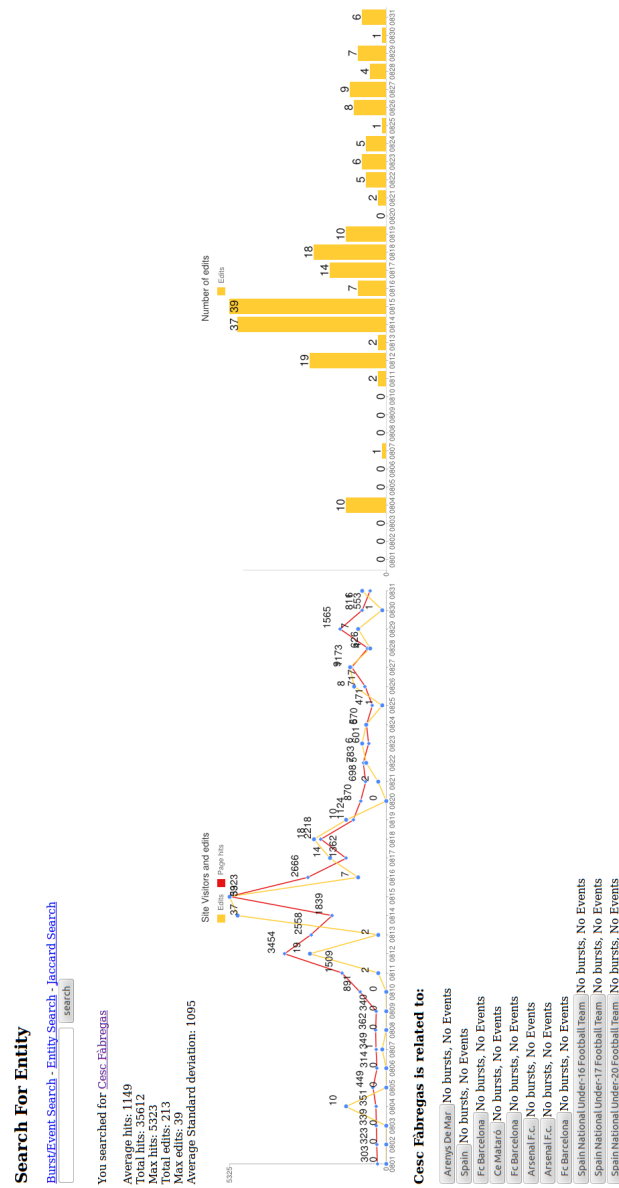


Figure 8.3: Search for Entity - Example View - Cesc Fàbregas

8.2.1 Redirects

The redirect part executes the query on the 'Redirect' table, with the search string appended an underscore. This indicates that this is the entire search string.

In Figure 8.4 there is a demonstration of how the redirect works. The code behind it consists of an if-sentence that will show the button if there exist any row in the table exactly matching the search string. In Figure 8.5 the redirect has been executed, and here the search string does not have any redirect matches, therefore the button is not visible.

Search For Entity

[Burst/Event Search](#) - [Entity Search](#) - [Jaccard Search](#)

Did you mean

You searched for [Cesc Fabregas](#)

Figure 8.4: Possible Redirect

Search For Entity

[Burst/Event Search](#) - [Entity Search](#) - [Jaccard Search](#)

You searched for [Cesc Fàbregas](#)

Figure 8.5: Redirected and No Further Redirects

8.2.2 Statistics

The first line in the statistics is the 'You searched for [entity]' and contains a link to the Wikipedia page for that entity. The rest of the statistics in the view is composed by using the values fetched from the statistic column family in the 'Pagecount' table, Table 5.1. However, there are two values that are not fully represented in the table, the average value and standard deviation, since they have to be calculated. The formulas deployed to obtain these values are shown in Equation 8.3 and 8.4. For an example view, see Figure 8.6

$$\sigma = \sqrt{\frac{\text{absoluteDeviation}}{\text{files}}} \quad (8.3)$$

$$\mu = \frac{\text{total}}{\text{files}} \quad (8.4)$$

You searched for [Cesc Fàbregas](#)

Average hits: 1149
 Total hits: 35612
 Max hits: 5323
 Total edits: 213
 Max edits: 39
 Average Standard deviation: 1095

Figure 8.6: Statistics for Entity

8.2.3 Graph Plotting

The graph plotting is implemented using *googlechartphplib*¹. In Figure 8.3, we can see two different graphs. A line chart containing both edit counts and page views, and a bar chart that contains only edit data. The reason for having two plots is to make all data fully visible for the user. The double plot graph has been tried out both with and without autoscale. It looks a bit disorganized when not using autoscale, but the graphs give a more and better information. Autoscaling makes the graphs relative to one another. As the page view generally is much higher than the edit count, autoscaling makes it harder to interpret events from bursts - as it is difficult to distinguish the edit values. See Figure 8.7 and 8.8

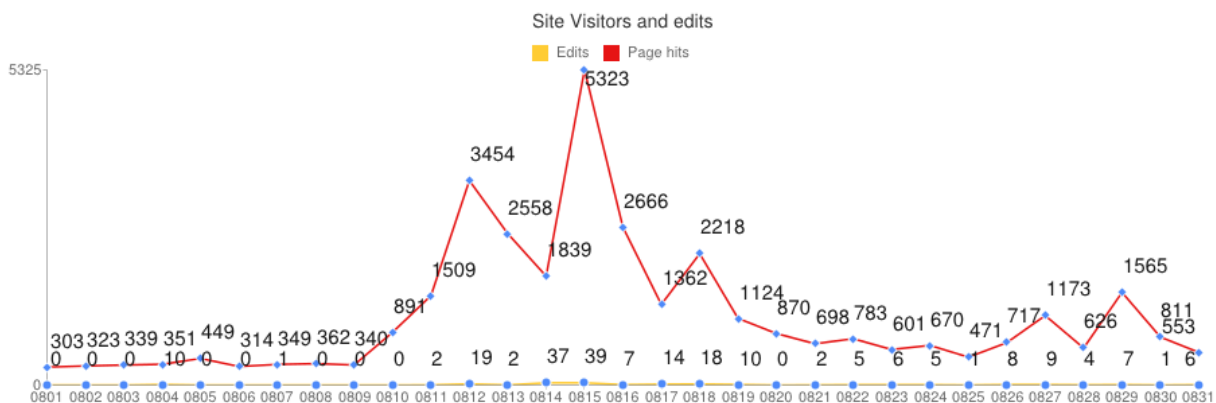


Figure 8.7: Double Plot Graph - Autoscale - Cesc Fàbregas

¹<http://code.google.com/p/googlechartphplib/>

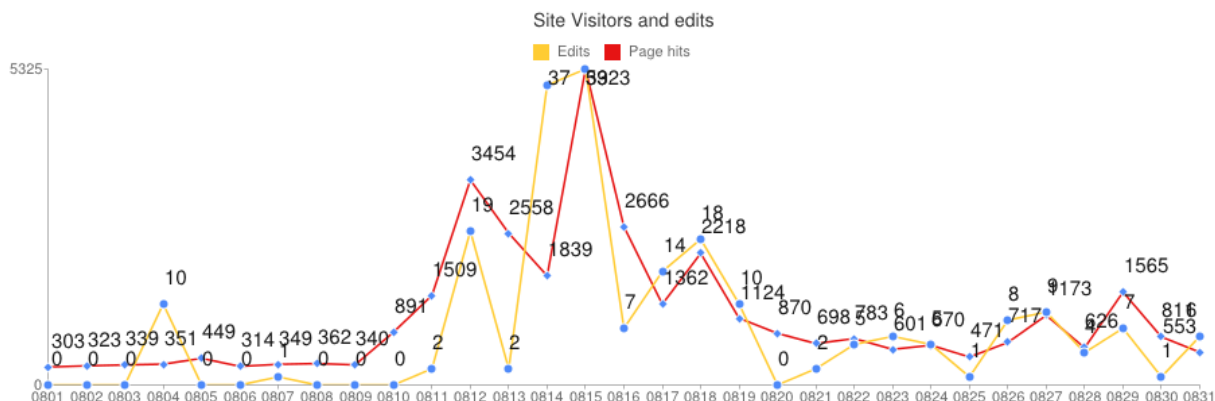


Figure 8.8: Double Plot Graph - No Autoscale - Cesc Fabregas

The extraction of page hits from HBase is conducted in the same way as all other queries. It queries the 'daily' column family for rows in the 'Pagecount' table. To put the values into the graph is fairly straightforward. Insert the page view values into a list and send them to the graph. The insertion of the page edit values is a bit more tricky, because some of the dates lack edit entries. The way this is solved is to insert a zero where entries do not exist, see Algorithm 7. The edit dataset created from this operation is used for both the line chart and the bar chart. In Figure 8.9 we show an example of how a bar chart looks.

Algorithm 7 Creation of Edit List for Plots

```

1: count = 0
2: for i = 0 → size of dates do
3:   for j → size of editDates do // editValues and editDates are of the same size
4:     if editDates[j] equals dates[i] then
5:       Increment count
6:       editValuesGraph add editValues[j]
7:     if count = i AND j = (size of editValues - 1) then
8:       Increment count
9:       graphEditValues add 0

```

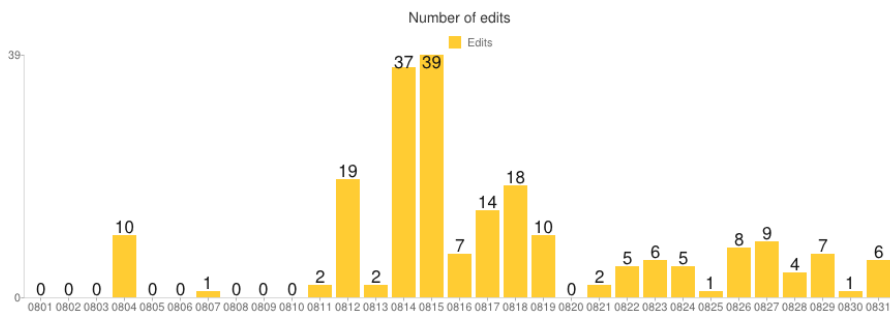


Figure 8.9: Bar Chart - Edit Count - Cesc Fabregas

8.2.4 Events and Bursts for Searched Entity

This part of the view is only displayed if there exist a burst or an event. This is the first place the redundant storing of the bursts table comes to use. The redundancy leads to a reduction in time usage for this part if compared to a non-redundant database. If there were no redundancy the system first would have to get all the bursts and events, and then for each entry in the burst table split the RowKey to get hold of the entity name, and further find the exact matches between the entity and the name of the entry in the burst table. To fetch events and bursts for the searched entity, we query the 'EventEntity' table two times. First to get hold of possible bursts, then to get hold of possible events. The values we retrieve are the number of hits and the Z-score, as well as edits for the events. In Figure 8.10 and 8.11 we show examples of how it looks when the entity experiences a burst or an event.

Burst dates for Tom Cleverley

20110822: Z-score: 12 Hit Count: 2412

Figure 8.10: Bursts for Searched Entity

Event dates for Bryan Ruiz

20110831: Z-score: 111 Page Edits: 51 Hit Count: 4311

Figure 8.11: Events for Searched Entity

8.2.5 Related Entities

The implementation of the related entities in the web application queries the 'Related Entities' table. Further, the related entities are printed by using a for loop to go through the result list from the query. If the query does not return any results, the page will display 'No Related Entities Found'.

Related Burst and Event

The related bursts/events are an elaboration of the related entities. This part uses the results gained from the related entity query, to again query the 'EventEntity' table twice for each entity, one time for each column family, namely 'event' and 'burst'. As seen in Figure 8.12 it shows 'No burst' and 'No event', which means that both the queries returned null. If something is returned, it will be displayed as 'Entityname Burst: Date Event: Date'. In addition there has been added a button for each related entity to give the possibility go to their page which contains graphs, statistics and the rest of the displayed information. The buttons, events and bursts can be seen in Figure 8.12 and 8.13. From the figure, we can see that there is some

redundancy. This is due to entities having multiple entries in the Wikipedia Infobox - Figure 4.5.

Cesc Fàbregas is related to:

Arenys De Mar	No bursts, No Events
Spain	No bursts, No Events
Fc Barcelona	No bursts, No Events
Ce Mataró	No bursts, No Events
Fc Barcelona	No bursts, No Events
Arsenal F.c.	No bursts, No Events
Arsenal F.c.	No bursts, No Events
Fc Barcelona	No bursts, No Events
Spain National Under-16 Football Team	No bursts, No Events
Spain National Under-17 Football Team	No bursts, No Events
Spain National Under-20 Football Team	No bursts, No Events

Figure 8.12: Related Entities without Burst or Events - Cesc Fàbregas

Bryan Ruiz is related to:

Fulham F.c.	No bursts, Event: 20110831
L.d. Alajuelense	No bursts, No Events
K.a.a. Gent	No bursts, No Events
Fc Twente	Burst: 20110816 No Events
Fulham F.c.	No bursts, Event: 20110831
Costa Rica National Football Team	No bursts, No Events

Figure 8.13: Related Entities with Burst and Events - Bryan Ruiz

8.3 Jaccard Similarity Search

This view has the same structure as the 'Entity Search' view, where the user can search for an entity. The possibility of redirects are also included in this view too. The results displayed is fetched from the JaccardScore table. For each row this view gets from the JaccardScore table, it sends a query to the Pagecount table to get hold of the daily values for the graphs. This is also done for the searched entity. A problem concerning the graph library emerged when this was implemented. The problem is that the chart library supports only 6 data lines. We worked

around this by using multiple charts in this view. Below the charts the results from the first query is displayed in a textual form like this: Entity, Jaccard Score: $x.x$, where x is a digit. An example view is shown in Figure 8.14.

Search For Jaccard Similarities

[Burst/Event Search](#) - [Entity Search](#) - [Jaccard Search](#) -

You searched for [Tim Cook](#)

Jaccard Index and Graph

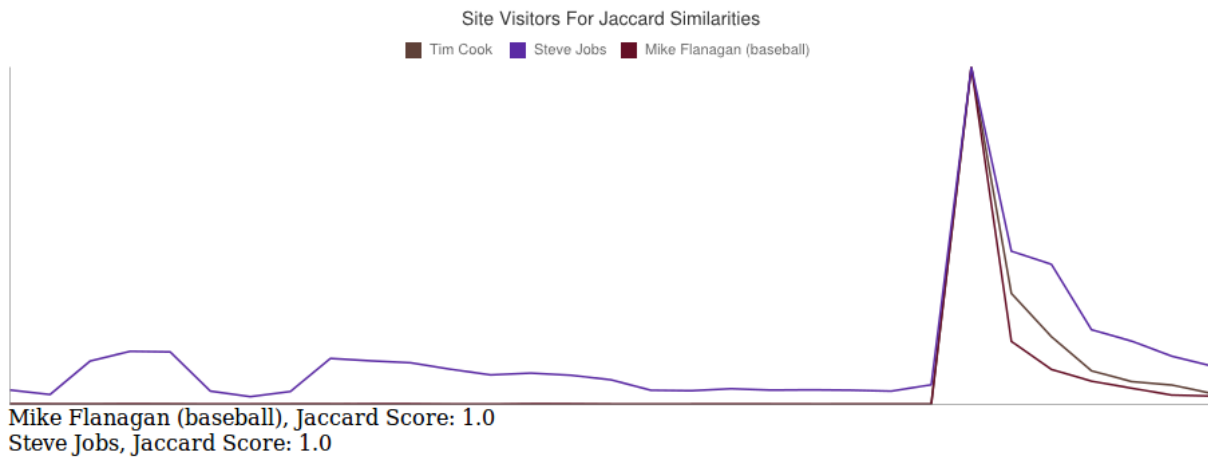


Figure 8.14: Example View - Jaccard Similarity Search - Tim Cook

Chapter 9

Examples

In this chapter we show examples of what the system can do. This will be presented using screenshots from the user interface, similar to some of the figures in Chapter 8. We will perform a case study on football transfers in August 2011, except Section 9.6, where the DARPA's Falcon Project is used. In addition to the screenshots from the web interface, there will be additional screenshots of Google searches that will explain the bursts and events. All examples are generated using the regular Z-scores, except in Section 9.5 where the results are based on the Modified Z-score because of a too low amount of bursts and events for the regular Z-score. For a quick recap, if the Z-score is above 10 and the Daily Page view exceed 250, we have a burst. In addition, if the Daily Page edits are above 20, we classify this as an event. In Table 9.1 the columns displayed in the different burst figures are explained, and in Table 9.2 the event columns are explained.

Entity	Page Views	Z-score	Burst level	Link to graphs	Google Search
--------	------------	---------	-------------	----------------	---------------

Table 9.1: Explanation of Burst Columns

Entity	Z-score	Edits	Page Views	Burst level	Link to graphs	Google Search
--------	---------	-------	------------	-------------	----------------	---------------

Table 9.2: Explanation of Event Columns

9.1 Burst Examples

In this section we present three football players that experience a burst during August 2011, namely Tom Cleverley, Mikel John Obi and Kolo Touré.

In the following three figures, Figure 9.1, 9.2 and 9.3, we show that Tom Cleverley is marked as bursting on 22 August 2011. This day there were 2412 views on this page, which gave a Z-score of 12 for the window spanning from 11 to 21 August. In Figure 9.2, the burst is the highest peak. However, there is another peak in this graph. This is not registered since the sliding window needs to be full before any calculations are performed - in our case, the 10 first dates are unable to burst or be an event, as the sliding window is still being filled. Figure 9.3

shows the Google search results when searching for the date combined with Tom Cleverley. This search confirms the reason for this burst; he played a game for Manchester United F.C. against Tottenham Hotspur F.C.

Survivor: South Pacific	430	15	1	Survivor: South Pacific	Google
Terrelle Pryor	2730	13	1	Terrelle Pryor	Google
Tilly Devine	670	12	1	Tilly Devine	Google
Tlc (band)	511	18	1	Tlc (band)	Google
Tom Cleverley	2412	12	1	Tom Cleverley	Google
Tripoli International Airport	292	10	1	Tripoli International Airport	Google

Figure 9.1: Bursts - Tom Cleverley - 22 August 2011

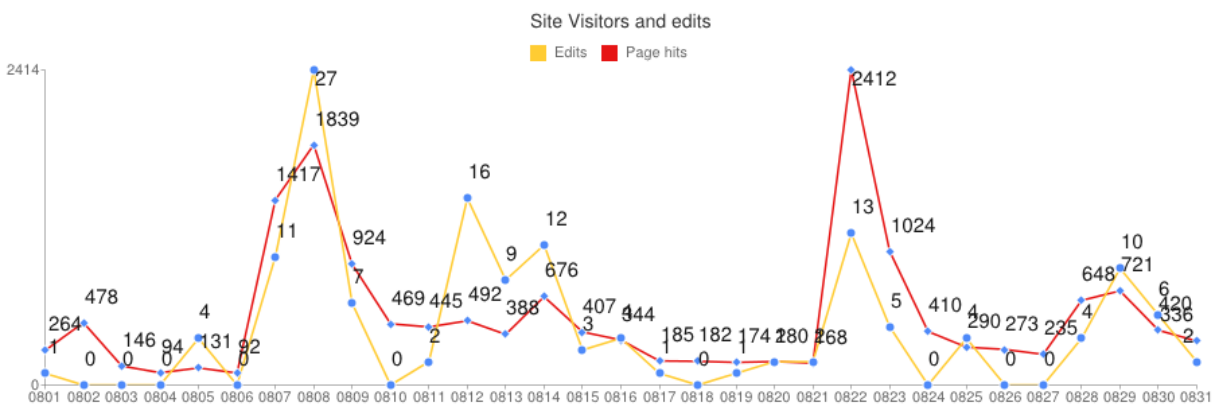
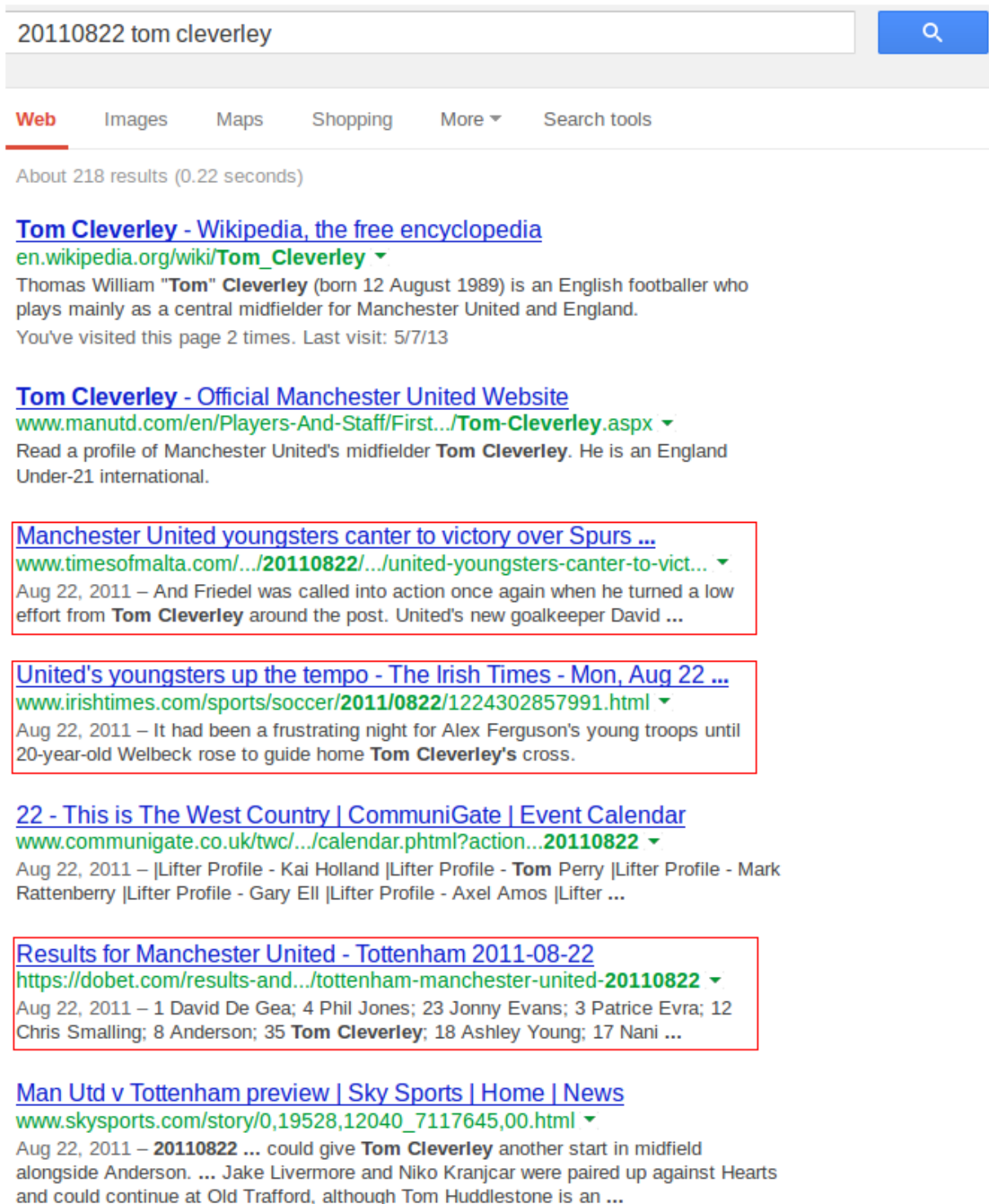


Figure 9.2: Line Chart - Tom Cleverley



20110822 tom cleverley

Web Images Maps Shopping More Search tools

About 218 results (0.22 seconds)

Tom Cleverley - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Tom_Cleverley ▼
Thomas William "Tom" Cleverley (born 12 August 1989) is an English footballer who plays mainly as a central midfielder for Manchester United and England.
You've visited this page 2 times. Last visit: 5/7/13

Tom Cleverley - Official Manchester United Website
www.manutd.com/en/Players-And-Staff/First.../Tom-Cleverley.aspx ▼
Read a profile of Manchester United's midfielder Tom Cleverley. He is an England Under-21 international.

Manchester United youngsters canter to victory over Spurs ...
www.timesofmalta.com/.../20110822/.../united-youngsters-canter-to-vict... ▼
Aug 22, 2011 – And Friedel was called into action once again when he turned a low effort from Tom Cleverley around the post. United's new goalkeeper David ...

United's youngsters up the tempo - The Irish Times - Mon, Aug 22 ...
www.irishtimes.com/sports/soccer/2011/0822/1224302857991.html ▼
Aug 22, 2011 – It had been a frustrating night for Alex Ferguson's young troops until 20-year-old Welbeck rose to guide home Tom Cleverley's cross.

22 - This is The West Country | CommuniGate | Event Calendar
www.communigate.co.uk/twc/.../calendar.phtml?action...20110822 ▼
Aug 22, 2011 – |Lifter Profile - Kai Holland |Lifter Profile - Tom Perry |Lifter Profile - Mark Rattenberry |Lifter Profile - Gary Ell |Lifter Profile - Axel Amos |Lifter ...

Results for Manchester United - Tottenham 2011-08-22
<https://dobet.com/results-and.../tottenham-manchester-united-20110822> ▼
Aug 22, 2011 – 1 David De Gea; 4 Phil Jones; 23 Jonny Evans; 3 Patrice Evra; 12 Chris Smalling; 8 Anderson; 35 Tom Cleverley; 18 Ashley Young; 17 Nani ...

Man Utd v Tottenham preview | Sky Sports | Home | News
www.skysports.com/story/0,19528,12040_7117645,00.html ▼
Aug 22, 2011 – 20110822 ... could give Tom Cleverley another start in midfield alongside Anderson. ... Jake Livermore and Niko Kranjcar were paired up against Hearts and could continue at Old Trafford, although Tom Huddlestone is an ...

Figure 9.3: Google Search - Tom Cleverley 22 August 2011

In Figure 9.4 we have marked two of the entities that are bursting 15 August, namely Mikel John Obi and Kolo Touré. Figure 9.5 and B.1 shows the graph and the Google search for Mikel John Obi. It is clearly shown where the burst is in the graph, Figure 9.5. The bursting day, Mikel had 351 hits which gave a Z-score of 23. The most interesting part is that this entity is not bursting because of footballing reasons, but because of the kidnapping of Mikel John Obi's father, see the marked Google results in Figure B.1.

Japanese Holdout	296	25	2	Japanese Holdout	Google
John Mellencamp	486	26	2	John Mellencamp	Google
Kolo Touré	542	29	2	Kolo Touré	Google
Lagaan	299	24	2	Lagaan	Google
Mikel John Obi	351	23	2	Mikel John Obi	Google
Naegleria Fowleri	689	20	2	Naegleria Fowleri	Google
Society Of Jesus	934	26	2	Society Of Jesus	Google
The Kliq	385	21	2	The Kliq	Google
Welcome Reality	426	23	2	Welcome Reality	Google

Figure 9.4: Bursts - Kolo Touré and Mikel John Obi - 15 August 2011

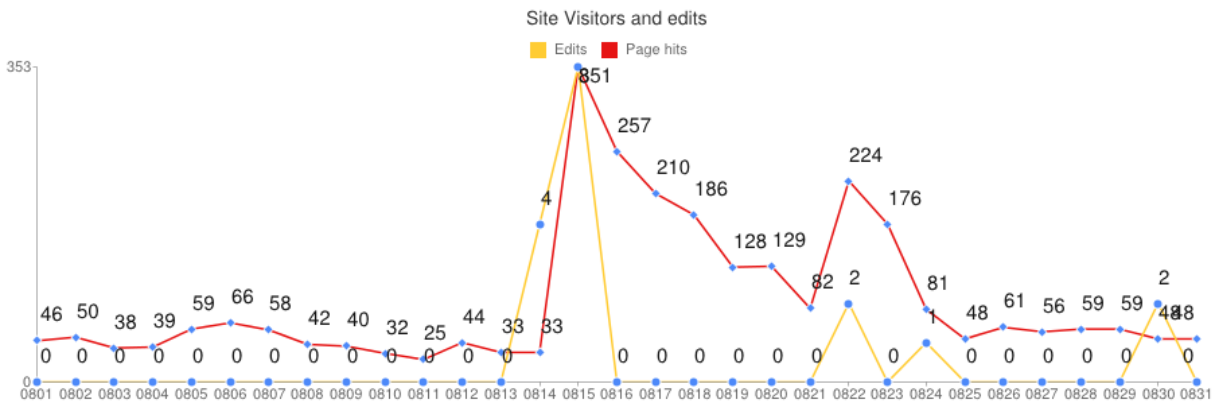


Figure 9.5: Line Chart - Mikel John Obi

The other person we have marked amongst the bursting entities in Figure 9.4, Kolo Touré, has only one peak during August 2011, see Figure 9.6. This day Kolo Touré's page had 542 page views which gives Z-score of 29. The Google search, shown in Figure B.2, gave the explanation that Kolo Touré started training again after a ban for drug abuse.

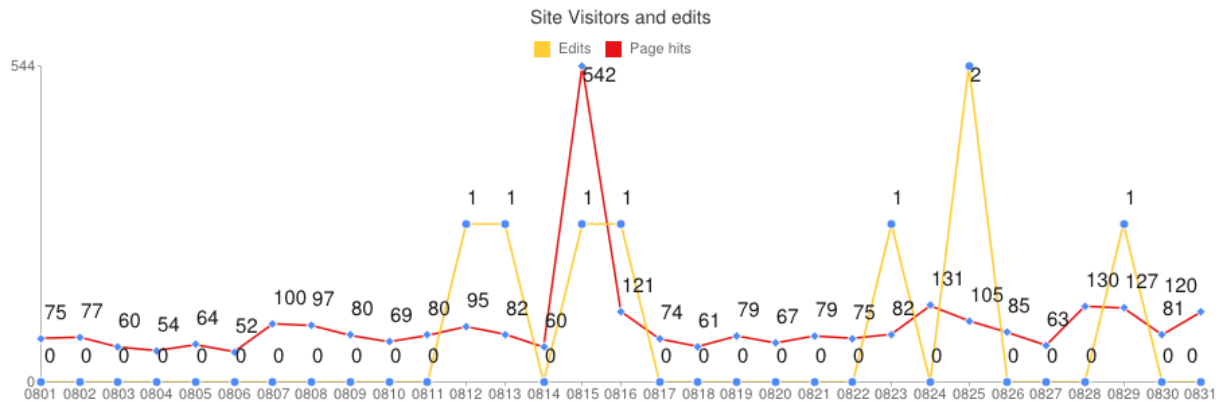


Figure 9.6: Line Chart - Kolo Touré

9.2 Event Examples

In this section four events will be presented and explained. The four entities selected as examples are Davide Santon, Bryan Ruiz, Mikel Arteta and Keisuke Honda.

Davide Santon is marked as an event 29 August 2011, as highlighted in Figure 9.7. We can see that the page had 999 views, 31 edits and a Z-score of 23. In the graph, Figure 9.8, we can see that 29 August is the start of a peak that ends on 30 August. 29 August is classified as an event, but the peak on 30 August is not. This will be discussed further in Chapter 10. The Google search, Figure 9.9, shows that Davide Santon was in Newcastle for a medical test, which is normal thing to do before transferring to a new club. On 30 August Davide Santon was sold from F.C. Internazionale of Milano to Newcastle United F.C.

2011 Mtv Video Music Awards	16	124	5511	1	2011 Mtv Video Music Awards	Google
Dancing With The Stars (u.s. Season 13)	19	22	1348	1	Dancing With The Stars (u.s. Season 13)	Google
Davide Santon	23	31	999	2	Davide Santon	Google
André Santos	306	89	3712	3	André Santos	Google

Figure 9.7: Events - Davide Santon - 29 August 2011

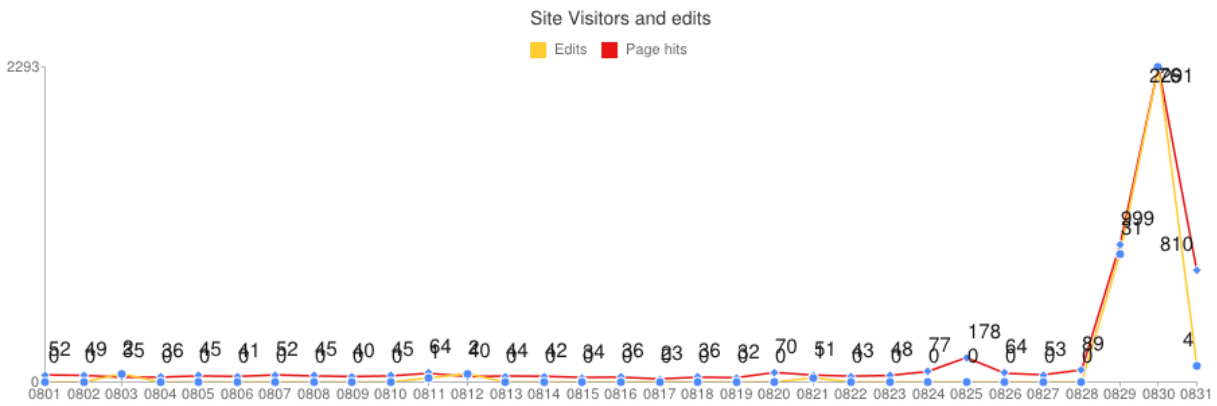



Figure 9.8: Line Chart - Davide Santon

20110829 davide santon 

Web Images Maps Shopping More ▾ Search tools

About 1,820 results (0.16 seconds)

[Davide Santon - Wikipedia, the free encyclopedia](#)
en.wikipedia.org/wiki/Davide_Santon ▾
Davide Santon (Italian pronunciation: [ˈdavide sanˈton]); born 2 January 1991 in Portomaggiore, Ferrara) is an Italian professional footballer who plays for ...

[Davide Santon - Goal.com](#)
www.goal.com/en/people/italy/22806/davide-santon ▾
Davide Santon – Current Season Statistics. League, Team, Games, Goals, Assists, YC, RC. England - Capital One Cup, Newcastle, 0, 0, 0, 0, 0. England - FA ...

[Davide Santon pulls out of Italy U21 squad with thigh injury](#)
www.newcastle-online.org > [Newcastle-Online](#) > [NUFC](#) > [Football](#) ▾
Aug 30, 2011 - 25 posts - 20 authors
Davide Santon pulls out of Italy U21 squad with thigh injury. ... http://www.nufc.co.uk/articles/20110829/santon-signs-on_2281670_2435425 ...

[More News - ESPN Soccernet - ESPN FC](#)
espnfc.com/news/more?section=italy&ver=ita?&...20110829... ▾
Aug 29, 2011 – **Davide Santon** undergoing Newcastle medical (Aug 29, 2011 11:00 GMT) Inter Milan full-back **Davide Santon** is undergoing a medical at ...

[More News - ESPN Soccernet](#)
soccernet.espn.go.com/news/more?section=England...20110829... ▾
Aug 29, 2011 – **Davide Santon** undergoing Newcastle medical (Aug 29, 2011 7:00 AM ET) Inter Milan full-back **Davide Santon** is undergoing a medical at ...

[El fichaje de Davide Santon por el Newcastle podría ser inminente ...](#)
www.fichajes.net > [Fichajes Italia](#) ▾ [Translate this page](#)
Aug 29, 2011 – Según ha publicado el diario inglés Daily Mail, el Newcastle United está a punto de cerrar la contratación de **Davide Santon**. El lateral italiano ...

Figure 9.9: Google Search - Davide Santon 29 August 2011

In the table of events in Figure 9.10, we have marked two entities, Bryan Ruiz and Mikel Arteta. Both of them are contained in burst level 3 on 31 August, which indicates that the Z-score for their respective sliding window is above 30. It is worth mentioning that nine out of the eleven entities displayed in Figure 9.10 are football players. For eight of them it is easy to find a legitimate reason for the event. Keisuke Honda is the ninth football player that was marked as an event. This will be elaborated later on in this section.

Bryan Ruiz had 4311 page views during 31 August, which gave a Z-score of 111. In addition the page had 51 edits, which indicates that something happened. From the graph, Figure 9.11, it can be seen that 31 August has an enormous explosion in page views. However, on 29 August the page had 21 edits which is above the threshold of marking an entity as an event. This can be a sign of many rumours about this player. When looking at the Google search, Figure B.3, the third result confirms that something concrete happened, Bryan Ruiz was sold from FC Twente to Fulham F.C. on 31 August 2011.

Bryan Ruiz	111	51	4311	3	<input type="text" value="Bryan Ruiz"/>	Google
Cameron Jerome	106	26	442	3	<input type="text" value="Cameron Jerome"/>	Google
Cody Mcdonald	48	28	513	3	<input type="text" value="Cody Mcdonald"/>	Google
David N'gog	122	43	1665	3	<input type="text" value="David N'gog"/>	Google
Denis Stracqualursi	229	141	1383	3	<input type="text" value="Denis Stracqualursi"/>	Google
Don Valley Parkway	367	76	1478	3	<input type="text" value="Don Valley Parkway"/>	Google
Eduardo Gonçalves De Oliveira	184	29	938	3	<input type="text" value="Eduardo Gonçalves De Oliveira"/>	Google
Jack The Ripper	129	30	5076	3	<input type="text" value="Jack The Ripper"/>	Google
Jermaine Beckford	71	26	897	3	<input type="text" value="Jermaine Beckford"/>	Google
Keisuke Honda	44	28	882	3	<input type="text" value="Keisuke Honda"/>	Google
Mikel Arteta	32	251	1907	3	<input type="text" value="Mikel Arteta"/>	Google

Figure 9.10: Events - Bryan Ruiz and Mikel Arteta - 31 August 2011

Mikel Arteta is another football player that was involved in a transfer 31 August 2011. He was sold from Everton F.C. to Arsenal F.C. The Wikipedia page had 1907 views, which gave a Z-score of 32, see Figure 9.10 and 9.12. The page also had 251 edits during that day, which is clear indication of an event. The Google search, Figure B.4, confirms the transfer.

In Figure B.5, we have a classic example of a rumour started by a newspaper, Twitter or similar. This day Keisuke Honda, according to the Google search, was strongly linked to Arsenal F.C. This is marked as an event, which it really is not. The explanation for this is probably that it was the last day of the transfer window, some fans were desperate for signings, and edited the Wikipedia page. This is a prime example of edit vandalism [13].

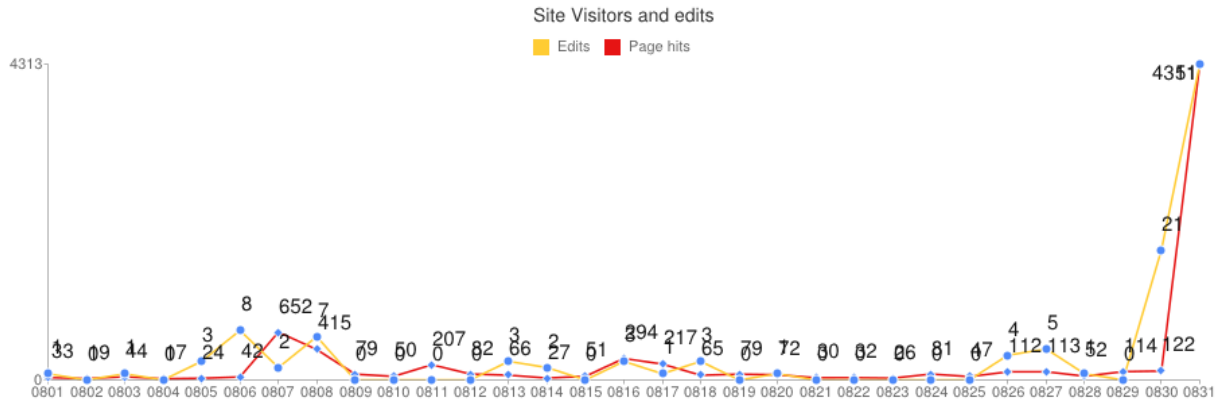


Figure 9.11: Line Chart - Bryan Ruiz

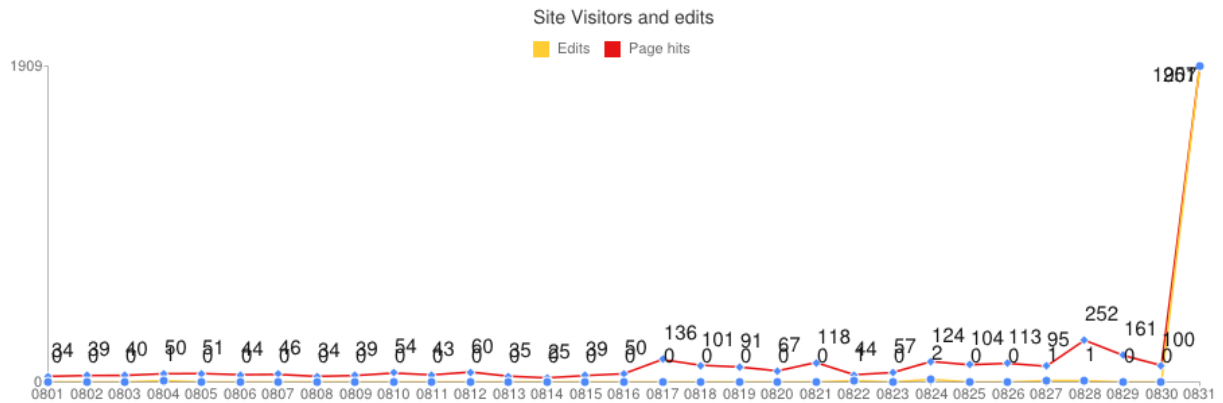


Figure 9.12: Line Chart - Mikel Arteta

9.3 Related Events and Bursts

One of the earlier examples has a related event. In Figure 9.13, we see that FC Twente is bursting at 15 August, and Fulham F.C. has an event on 31 August. The interesting part here is the event, since Bryan Ruiz was sold to Fulham F.C. that date and, according to the Google search in Figure B.6, this was the reason for the event.

Bryan Ruiz is related to:

Fulham F.c.	No bursts, Event: 20110831
L.d. Alajuelense	No bursts, No Events
K.a.a. Gent	No bursts, No Events
Fc Twente	Burst: 20110816 No Events
Fulham F.c.	No bursts, Event: 20110831
Costa Rica National Football Team	No bursts, No Events

Figure 9.13: Related Event and Burst - Bryan Ruiz

9.4 No Detection Examples

In this section we will present some graphs that ideally should have been detected as bursts or events. Two of the biggest transfers during the transfer window in August 2011 was Cesc Fàbregas and Samir Nasri. None of these are marked as events or bursts during the time period that has been analysed. The main reason for this is that the page views - see Figure 9.14 and 9.15 - have a gradual increase, and these gradual increments increase the sliding windows' standard deviation by a large amount, making it difficult to reach a high Z-score.

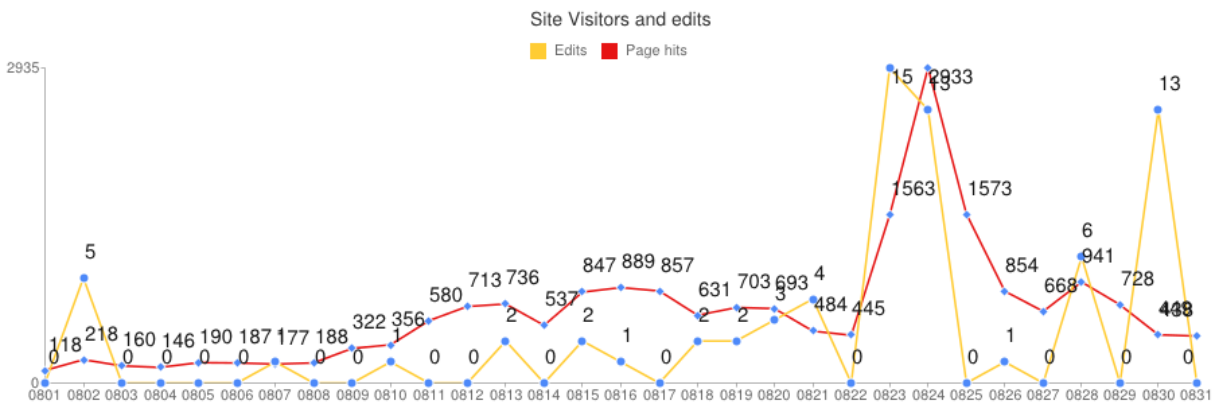


Figure 9.14: Line Chart - Samir Nasri

9.5 Jaccard Similarity

The results for the Jaccard similarity were disappointing, but we managed to find some interesting results. The results were lacking mainly because of a small time window. We expect the results to improve if this is tested out on a larger document set. As shown in Chapter 8 the system paired Steve Jobs and Tim Cook - former and current CEO of Apple Inc. - through the

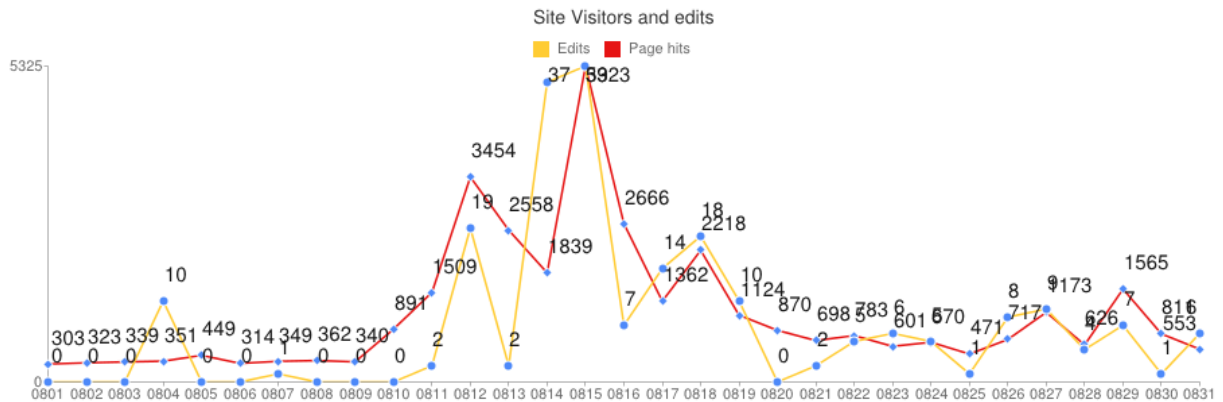


Figure 9.15: Line Chart - Cesc Fàbregas

use of Jaccard similarities, which is a good match. It can also be seen that both these persons have the same tendency in the graph, see Figure 9.16.

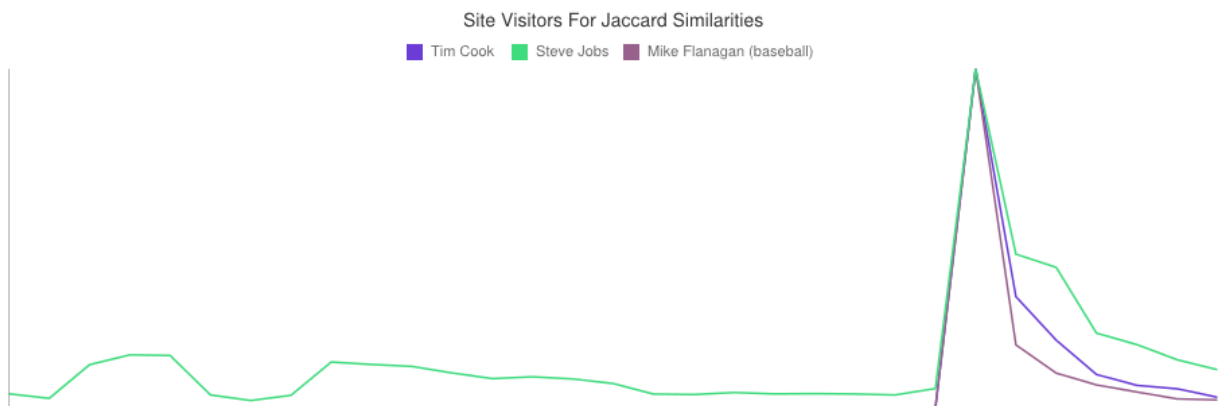


Figure 9.16: Line Chart - Jaccard Similarity - Steve Jobs, Tim Cook, Mike Flanagan

In Figure 9.17 we display an example of how this feature can produce wrong data. We check retrieve all the entities that have a high similarity to Davide Santon. None of the entities are in anyway related, or can be spoken about in the same sentence, however they have the same burst dates and page view tendency.

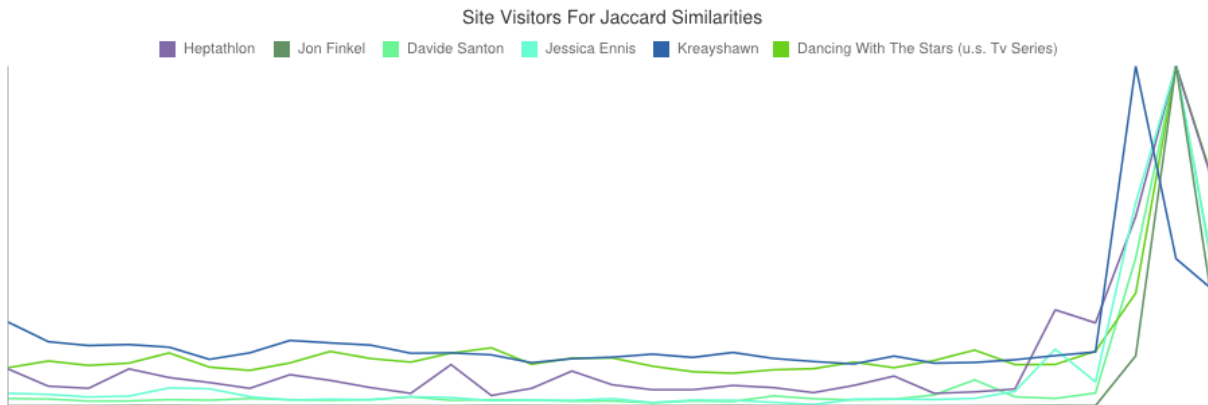


Figure 9.17: Line Chart - Jaccard Similarity - Davide Santon

9.6 Grouping of Entities

Another interesting part discovered when experimenting with the results, was that some entities should have been related or grouped together. One example for this is the DARPA's Falcon Project, shown in Figure 9.18 and 9.19. All the marked entities are bursting 11 August, related to the same project, and the graphs provided in Figure C.1, C.2, C.3 and C.4 shows that all these entities have the same tendency in terms of the shape of the graph. This indicates that the usage of Wikipedia infoboxes not necessary is the best way to extract related entities, and that a grouping functionality could have been another feature to explore. All these entities could not have been snapped up by the Jaccard similarity because of a different and low amount of bursts and events, at least not as it is implemented today.

Darpa	675	18	1	Darpa	Google
Darpa Falcon Project	6494	15	1	Darpa Falcon Project	Google
Frances Ford Seymour	601	12	1	Frances Ford Seymour	Google
George Soros	1987	15	1	George Soros	Google
Jennifer Love Hewitt	1903	13	1	Jennifer Love Hewitt	Google
Khudiram Bose	253	15	1	Khudiram Bose	Google
Lamborghini Reventón	442	14	1	Lamborghini Reventón	Google
Madagascar	1128	10	1	Madagascar	Google
Mark-paul Gosselaar	1757	14	1	Mark-paul Gosselaar	Google
Minotaur Iv	325	13	1	Minotaur Iv	Google
Miranda Cosgrove	1910	12	1	Miranda Cosgrove	Google
Paddy Mccourt	655	13	1	Paddy Mccourt	Google
Paul Ince	260	15	1	Paul Ince	Google
Portable Document Format	1156	19	1	Portable Document Format	Google
Rockwell X-30	336	17	1	Rockwell X-30	Google
Rory Mcilroy	799	13	1	Rory Mcilroy	Google
Samsung Galaxy S li	1212	17	1	Samsung Galaxy S li	Google
X-41 Common Aero Vehicle	307	16	1	X-41 Common Aero Vehicle	Google

Figure 9.18: DARPA's Falcon HTV-2 Hypersonic Aircraft - 11 August 2011

Htv-2	553	49	3	Htv-2	Google
-------	-----	----	---	-------	------------------------

Figure 9.19: DARPA's Falcon HTV-2 Hypersonic Aircraft - 11 August 2011

Chapter 10

Results and Analysis

This chapter presents results and contains analysis of how the system performs. It will discuss problems uncovered in the case study on football transfers from Chapter 9, such as the entities which should have been bursting, and grouping of entities. In addition we explain why related entities do not experience bursts or events when it is expected. The results will be presented using two different calculations of the Z-score in the sliding window. We will then set these results up against each other in order to determine which of the methods that gives the best results.

10.1 Football Transfers

The findings from the case study of football transfers is presented in the following sections. The two different methods of data extraction, namely 'Z-score' and 'Modified Z-score', were explained in Chapter 6. The 'Z-score' use a sliding window size of 10 days and the 'Modified Z-score' use a sliding window size of 11 days. Finding football transfers from the bursts and events detected have been done through manual inspection.

10.1.1 Burst and Event Precision

In this section we present tables that will show the precision of the system and we present a bar chart that summarises our findings. The chart and tables contain five categories. 'Performance' is a category that contains entities that have been involved in something that is related to football, for example playing in a game or scoring a goal. 'Rumour' is the category where there exists a newspaper article or similar that connects the entity to another club. 'Rumour - Transfer' means that the entity experiences an event or burst ahead of the actual transfer. 'Transfer' are entities that experience an event or a burst on the date of the announcement of the transfer, and 'Other' means that the entity had a burst or an event with no reference to football.

Z-scores - Events

We can see from Figure 10.1 and Table 10.1 that transfers represent almost 50% of the events extracted. In addition, around 25% are also transfers, but have been marked as an event before

the transfer actually happened. From this we can derive that 75% of the extracted football players that experience an event during August 2011 can be categorized as a transfer. The entities in the 'Performance' part of this table are high profiled football players.

Extracted Events - Z-score		
Type	Number	Proportion
Other	0	0.000
Performance	11	0.204
Rumour	1	0.019
Rumour - Transfer	16	0.296
Transfer	26	0.481
Total	54	1

Table 10.1: Extracted Events - Z-score

Z-scores - Bursts

From Figure 10.1 and Table 10.2, it can be seen that the burst extraction consists of around 60% performances, which was expected. When a person does something good or bad many people usually check that person's Wikipedia page. In addition, the players extracted in the 'Transfer' part are mainly less profiled players with a low edit count. Many of the 'Performance' bursts are detected at the same day as a game, or the day after. The 'Rumour' category in Figure 10.1, is close to the same amount in percentages for both bursts and events. This show that less serious newspapers and similar that create rumours do not have a big impact on the Wikipedia page views. However the 'Rumour - Transfer' section is a notable part of both tables. This part shows that many of the rumour bursts and events end up in a transfer after a little time.

Extracted Bursts - Z-score		
Type	Number	Proportion
Other	1	0.009
Performance	72	0.615
Rumour	3	0.026
Rumour - Transfer	11	0.094
Transfer	30	0.256
Total	117	1

Table 10.2: Extracted Bursts - Z-score

Z-scores - Total

Figure 10.1 contains the extracted data from both the events and bursts extracted from the system. It can be seen from Table 10.3 that the system extracts about 50% 'Performance' entities and about 50% 'Transfer' or future 'Transfer' entities.

Extracted Bursts and Events - Z-score		
Type	Number	Proportion
Other	1	0.006
Performance	83	0.485
Rumour	4	0.023
Rumour - Transfer	27	0.158
Transfer	56	0.327
Total	171	1

Table 10.3: Extracted Bursts and Events - Z-score

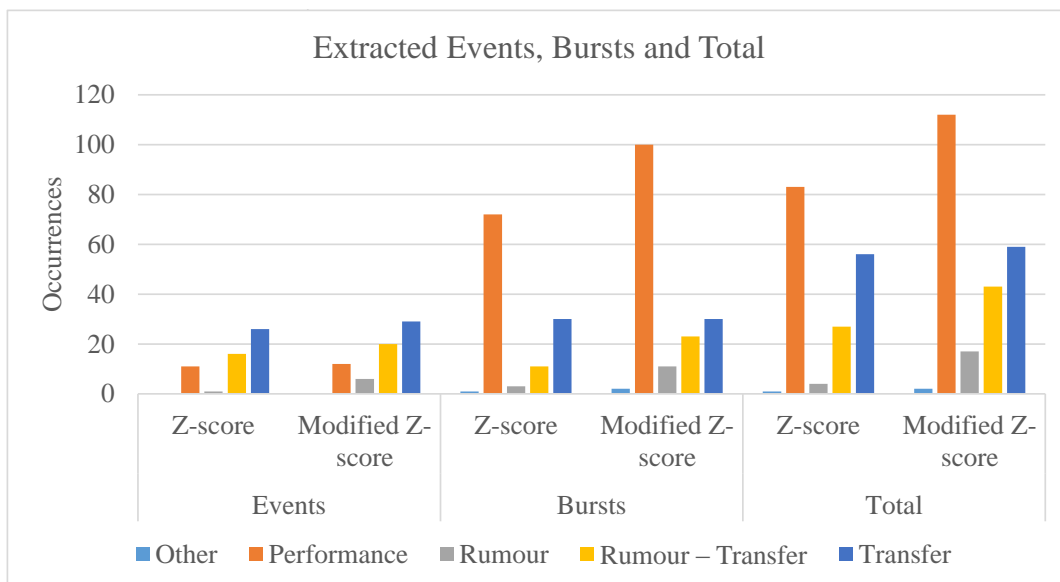


Figure 10.1: Results after Manual Inspection of Bursts and Events

Modified Z-scores - Events

The events extracted with the usage of 'Modified Z-score' are presented in Table 10.4 and Figure 10.1. From the table it can be derived that 73% of the events were transfers. In addition, from the rumours and rumour - transfers we can calculate that 77% ($20/(20+6)$) of the rumours turns into transfers.

Extracted Events - Modified Z-score		
Type	Number	Proportion
Other	0	0.000
Performance	12	0.179
Rumour	6	0.089
Rumour - Transfer	20	0.299
Transfer	29	0.433
Total	67	1

Table 10.4: Extracted Events - Modified Z-score

Modified Z-scores - Bursts

In Table 10.5 and Figure 10.1 the extracted bursts are presented. According to this, the system mainly extracts entities that are in the category 'Performance', 60%. The transfers extracted here are mainly low profile entities, who generally have fewer edits, and they are therefore unable to be classified as events. The 'Rumour - Transfer' category contains in this case some of the 'Transfer' entities from Table 10.4. This means that many entities experience bursts from rumours before the actual transfer happens. The 'Performance' category contains some of the same entities because some of them bursts after each game they are involved in.

Extracted Bursts - Modified Z-score		
Type	Number	Proportion
Other	2	0.012
Performance	100	0.602
Rumour	11	0.066
Rumour - Transfer	23	0.139
Transfer	30	0.181
Total	166	1

Table 10.5: Extracted Bursts - Modified Z-score

Modified Z-scores - Total

Under the total column in Figure 10.1 and Table 10.6 it is displayed what the 'Modified Z-score' extracts. In total it shows that about 50% of the extracted entities are contained in the 'Performance' category. The 'Transfer' and 'Rumour - Transfer' parts for the 'Modified Z-score' amounts to about 45%, which is 5% lower compared to the 'Z-score'. The last 8% are contained in the 'Rumour' and 'Other' categories.

10.1.2 BBC List

In the next paragraphs we will use a list of all football transfers in England in the period of 11 to 31 August 2011. In this experiment all the transfers from the list in Appendix A.1 will be

Extracted Bursts and Events - Modified Z-score		
Type	Number	Proportion
Other	2	0.009
Performance	112	0.480
Rumour	17	0.073
Rumour - Transfer	43	0.185
Transfer	59	0.253
Total	233	1

Table 10.6: Extracted Bursts and Events - Modified Z-score

examined for bursts and events for the dates on and around the transfer date. This data will be classified in four categories. One category for bursts, one category for events, and one for the entities that did not have a burst or an event. The final category 'Too Low', seen in Table 10.7, will not feature in the results because this is not really interesting data, as the system is unable to mark the entities as bursts at all.

The first examination of this list showed that 127 entities did not have enough page views to climb over the threshold of 250 views per day and they are therefore categorized as irrelevant, and not taken into account when the percentage of the findings are presented.

Z-scores

When looking at the chart in Figure 10.2 and the numbers under the 'Refined Proportion' in Table 10.7, it shows that the system extracts close to 66% of the transfers possible to detect. 45% of the extracted entities are events, 21% bursts. Some of the transfers that were not detected fall into the category of 'Z-score' problems, which will be explained in Section 10.6. Many of the transfers that were found as bursting were low profile players. This shows the problem related to the usage of a naïve threshold in the edit count, which means that an entity needs above a specific amount of edits to be categorised as an event

BBC List - Z-score			
Type	Number	Proportion	Refined Proportion
Too Low	127	0.655	0
Not Found	23	0.119	0.343
Event	30	0.155	0.448
Burst	14	0,072	0.209
Total	194	1	1

Table 10.7: BBC List - Z-score

Modified Z-scores

Here the whole list provided by BBC was examined with the usage of the 'Modified Z-score'. We can see from Table 10.8 and Figure 10.2 that the system extracts about 56% of the transfers as events, and around 20% as bursts, meaning that the system manages to extract over three quarters of transfers possible to detect in the BBC list.

BBC List - Modified Z-score			
Type	Number	Proportion	Refined Proportion
Too Low	122	0.645	0
Not Found	16	0.085	0.239
Event	38	0.201	0.567
Burst	13	0.069	0.194
Total	194	1	1

Table 10.8: BBC List - Modified Z-score

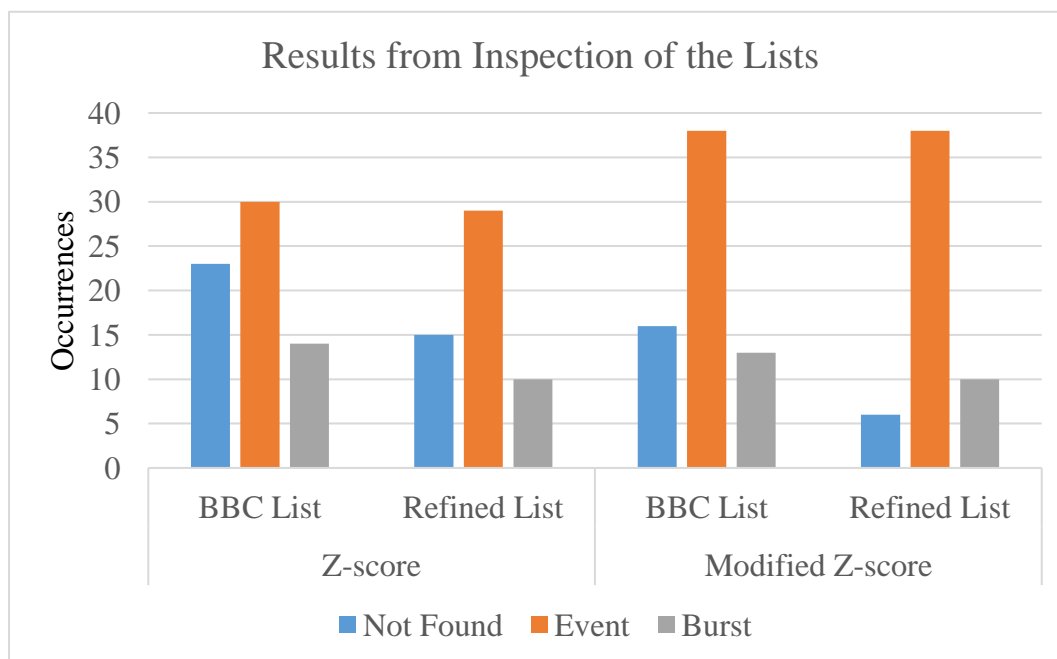


Figure 10.2: Results from the List Inspection

10.1.3 Refined List

In this section the refined list presented in Appendix A.2 is examined to uncover if there exist some difference between the two methods, 'Z-score' and 'Modified Z-score'. This list is shortened by the entities that were in the 'Too Low' category when conducting the experiment on the BBC list, Section 10.1.2. In addition we manually removed the entities that we found irrelevant, such as transfers between clubs that play in the lower divisions and some loan transfers from big clubs to lower division clubs.

Z-scores

Compared to the BBC-list the results changed slightly. Table 10.9 show how the system performed on this list, and compared to the whole list it extracts around 10% more of the transfers, which is good since we want a system that is to be able to extract the big transfers, not the transfers between lower division clubs. The percentage of transfers that were found was close to 72%, which is 8% higher compared to the results from the BBC-list with the usage of Z-score. In Figure 10.2, the same numbers as in Table 10.9 are shown in a graphical illustration.

Refined List - Z-score		
Type	Number	Proportion
Not Found	15	0.277
Event	29	0.537
Burst	10	0,185
Total	54	1

Table 10.9: Refined List - Z-score

Modified Z-scores

From the refined list, the 'Modified Z-score' performed better than the 'Z-score'. From Table 10.10 we can see that 70% of the transfers were marked as an event, and about 88% of the transfers were found by the system - either as an event or as a burst. This means that the system is able to find close to nine out of ten transfers. A graphical overview is contained in Figure 10.2.

Refined List - Modified Z-score		
Type	Number	Proportion
Not Found	6	0.111
Event	38	0.704
Burst	10	0,185
Total	54	1

Table 10.10: Refined List - Modified Z-score

10.1.4 Time Delay

The purpose of this experiment was to see if there is a certain lag in the Wikipedia page views. For example, to see if an entity receive a spike in page views the day after a completed transfer. However, the number of entities that spiked too late were so few that the delay was not interesting to look at. A more interesting thing was when the system found rumours that led to transfers. In Figure 10.1 the two categories called 'Rumour' and 'Rumour - Transfer' would in a real time system be categorised as the same. Since these tests use old data we are able to verify the data and the results from this is quite interesting. According to the numbers in Table 10.1 and 10.2 four entities in total were categorized as 'Rumour' and 27 entities in the 'Rumour - Transfer'. When calculating the percentage of rumours that turn into a transfer it shows that 87% of them turns into a transfer, which is quite astonishing. This number is a bit lower for the 'Modified Z-score', 71%, but this is still a great result. Overall, for both methods, at least 7 out of 10 extracted rumours leads to transfers. However it may be a bit imprecise to say this because many of the 'Rumour - Transfer' was detected one day in advance of the actual transfer, which in many cases already means that the transfer is a done deal.

10.2 Event Detection

In Chapter 1, we anticipated that the system would mainly detect transfers in the event category, but from the results presented in Section 10.1 it does not show a real big difference between bursts and events. The reason for this is the naïve usage of the edit history count, which is to set a static threshold. It is however a more challenging task to use the edit count wisely, as the amount of edits are not comparable to page views. This is due to a usually very low amount of edits. The usage of a naïve threshold works well in many circumstances, but falls short for the less popular entities.

10.3 Related Entities

As shown in Chapter 9, only one of the entities that had an event had a related event or burst at the same date. There are many reasons for this, but in the case of the Z-scores, the main reason is the big difference in the page views for the related entities. For example, when people enter a page, many of them enter one of the related entities too, and the related entity is linked to from more entities compared to the first entity. This leads to an irregular curve for the related entities with more than one popular linked up entity. This improved when the 'Modified Z-score' was applied and is one of the main problems concerning the usage of the Z-score, see Sections 10.6 and 10.7.

10.4 Jaccard Similarity

The Jaccard feature in this system works in a way, but it is indisputable that this feature needs a much bigger dataset to work in a good way. 21 days is too short for using the Jaccard similarity

index, but the implementation works and it manages to extract entities. Because of the small window of bursts it will in most cases group non related entities together. As demonstrated in Chapter 9, it groups Steve Jobs and Tim Cook with a baseball player. Putting Steve Jobs and Tim Cook together, with no other indications of any relations between them, is a good accomplishment. The baseball player is a pure coincidence.

10.5 Grouping Entities

In Section 9.6 we presented some entities that could have been grouped up. When speaking of grouping up entities we mean that all cities, football players and so forth, are grouped together. This would have been a more extensive thing to do compared to the current handling of related entities. An example of a way to do it is to add a tag for each entity to mark which category they belong to. Some of these groups could have been accomplished by using a DBpedia file that contains the types of each entity. For example 'person', 'city' and 'sportsTeamMember'. This would have solved the situation in many cases, but those Wikipedia articles that does not have any info box would not be included in this file, nor in the related entities. This would lead to processing the whole of Wikipedia corpus to get a decent system, and in this case this is an unrealistic task. It would also provide very general grouping. Other possibilities include examining the Wikipedia link graph, as is done in for example [6].

10.6 Z-scores

A problem uncovered while trying out the system was that entities with a stepwise increase in the daily page views not being marked as bursts when they should have, see Figure 10.3. The reason for this is that the standard deviation increases a lot when big numbers get introduced into a series of small numbers. The first step of a burst usually gets marked as a burst, but this increases the standard deviation by a big number, which makes it almost impossible for the second and third part of the burst to be marked as a burst. For the third part of a burst, at least for bursts like the one shown in Figure 10.3, the page views are about to normalize again, hence it is not as important as the first and second part of the burst.

This could have been solved by giving the bursting value a more normalized value in the window, but if this gets carried out another problem arises. The problem is that if an entity's number of page views suddenly rise to a much higher amount, and the page views stabilise around this value, it would take the amount of days in the window to even out. The process of normalising values in this way is referred to as *smoothing*.

10.7 Modified Z-scores

A thing we experienced while testing the system using the 'Modified Z-score' method, was that it extracted bursts around the peaks of the graph, unlike 'Z-score' (see Figure 10.3). This means

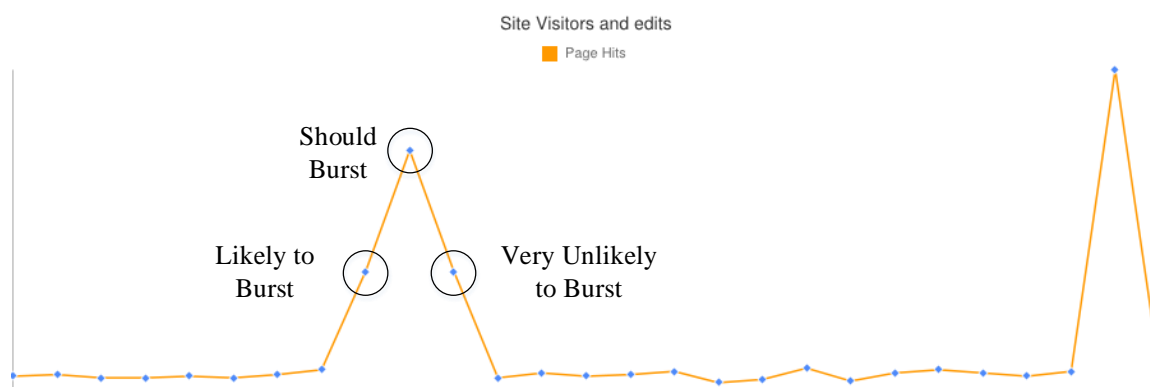


Figure 10.3: Illustration of Z-score Failure

that we are able to extract multiple bursts from the same window. However, it is discussable how much value this gives the system, as in many cases the bursts around a peak are unnecessary since the entity in most of the cases experience a burst while the graph is on the rise, and if this gives three or four bursts for the entity, we would say this decreases the precision of the system and increases the recall. This effect is enhanced by the fact that our system has no concept of burst lengths.

Another thing we experienced was that this method in some cases marked entities as bursting in rough graphs with small differences from the mean value for the entity. This is not a huge problem concerning the results, but it gives some irrelevant results which lowers the precision of this method.

10.8 Modified Z-scores Versus Z-scores

In Figure 10.4 the values from the examination of all bursts and events are presented, and in Figure 10.5 the results from the two different lists that we examine is presented.

There were no big differences concerning the number of events extracted for the methods, see Figure 10.4. However, the 'Modified Z-score' extracted a higher amount of events in each category. A thing that is worth mentioning is that the 'Z-score' method extracts very few rumours. The 'Modified Z-score' extracts six times as many.

In the burst extraction, the biggest difference is in the number of entities contained in the 'Performance' category, see Figure 10.4. The main reason for this is that the 'Modified Z-score' managed to extract bursts while the sliding window still contained an earlier burst, which the 'Z-score' mainly did not. The difference in the performance part mainly consists of entities that was bursting more than one time. Both methods extracts the same amount of transfers, but the 'Modified Z-score' extracts more rumours that becomes transfers.



Figure 10.4: Bar Chart - Event - Z-score Versus Modified Z-score

The overall chart, Figure 10.4, has one more column compared to the other charts. It shows that the 'Modified Z-score' extracts around 60 entities more, in total, as opposed to the 'Z-score' method. Half of this difference is contained in the 'Performance' category.

In Figure 10.5 the findings from the BBC list is presented. It shows that the two methods had a different amount of entities in the 'Too Low' category. The reason is that the 'Modified Z-score' starts at 12 August while the 'Z-score' starts at 11 August. All the transfers from 11 August was in the 'Too Low' category so it did not affect the rest of the results. From the figure it can be seen that the 'Modified Z-score' finds more entities, but there is not a high amount that separates the two methods in any of the categories.

The findings from the refined list is presented in Figure 10.5. From the figure we can derive that the 'Modified Z-score' achieves better results on event extraction, while there is no significant difference when looking at the burst extraction.

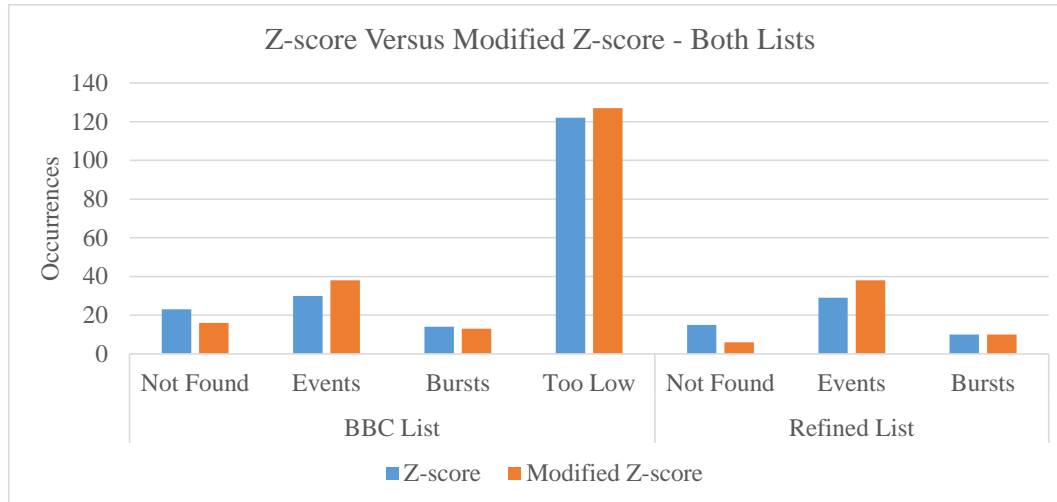


Figure 10.5: Bar Chart - BBC List - Z-score Versus Modified Z-score

10.9 Summary

From all the figures in this chapter, one might think that it would be easy to determine which of the methods that is the best one for extracting burst and events. The fact is that 'Modified Z-score' extracts more bursts in total, which can make this method slightly imprecise. The normal 'Z-score' is not able to extract bursts two days in a row, and in most cases it is not able to extract a new burst before the bursting date is out of the sliding window, which means that n days have to pass before a new burst can occur, where n is the size of the window.

For the experiments conducted on all the extracted bursts and events, both methods achieved some astonishing results related to rumours that turn into transfers. Both methods had above 80% accuracy in regards to rumours turning into transfers. The performance category was in this experiment a large part of the result set which was as expected, but we did not expect that such a big part of the events were categorized as performances. We did not expect that people would edit the entities as often when an entity experienced a performance burst. As mentioned in Section 10.2, the extraction is favouring popular entities, as they have a higher edit count. In this project the criteria to be marked as an event is to be experiencing a burst, and have an edit count above 20 during 24 hours. This is for many less popular entities very difficult to reach. This is with high probability the reason for the small differences between what the bursts and events contain. The probability for a transfer being marked as an event is higher than for it

being marked as a burst.

From the experiments on the two different lists, it is shown that both methods manage to extract between 65% and 88% of the transfers that is possible to detect in the lists. It is though the 'Modified Z-score' that achieve the best results in these tests.

For the Jaccard similarities the system needs a lot more bursts and events to be able to detect a better list of similar entities. However, the tables and back end solution for this is created for a scalable system so it will not be a problem inserting more data into the system to get better and more refined Jaccard similarities. Another thing related to Jaccard similarities is the grouping of entities. As described in Section 10.5, it would have been an idea to try and group entities since the related entities do not provide this feature, it only links the entity to the parent. The idea here is to group certain types together, for example put persons into a group, or even be more specific. The fast and easy solution is to use a DBpedia file, but this would not work for new entities, and entities without infoboxes. In addition, it is worth to mention that the Jaccard similarities favours the use of 'Modified Z-scores', as it generates a higher amount of bursts and events.

Taking all the above discussions into account, it shows that the two methods both have their strengths and weaknesses. The 'Modified Z-score' favours recall, and the 'Z-score' favours precision. Both have a decent performance and extract mainly the right entities. The main issue here is the definition and calculation of an event, which favours the popular entities. To get a better result set, the edit count history should not only be based on a naïve threshold. One should go through a form for calculation to get a more normalized value, to give all entities the possibility to climb over a certain value.

Chapter 11

Further Work

While the system fulfils all the requirements we have set for it, there are still multiple possibilities that would be interesting to work on in the future. The first is to transform the system from one that strictly works as a window to the past, to a system that provide up-to-date daily page view statistics and performs near real-time burst and event detection. The HBase schema does not need any modifications in order for this to work, but we need to implement suitable methods for real-time updating of the database. The system would also require functionality that allows it to automatically download and verify new Wikipedia page view files. A problem concerning a real-time application is the releases of the Wikipedia corpus dumps. These dumps are only released once a month, and these files are the source for the edit counts, which are essential for distinguishing events from bursts.

An obvious task is to test the system on a bigger data set. While we would fully expect the burst and event detection to extract the same type of results, the main benefactor of a bigger data set would be the comparison of Jaccard similarities. The experiments thus far are conducted on too small of a data set, and the Jaccard similarity comparison is unable to extract quality results.

A transformation of granularity, from storing information about days to a storing information about hours is also a possibility. This would show the user when an entity was bursting with a higher accuracy. However, changing the granularity in this way could also possibly increase the look-up times in the system, as 24 times as many rows would need to be fetched from HBase.

Another interesting thing to look into is to look at the size of edits, or in some way analyse the actual contents of the edits, as in [13]. Doing this would give us better opportunities when distinguishing events from bursts, as we could go away from the strictly naïve threshold we use today. It is reasonable to assume that larger events provide more context than smaller ones.

Grouping entities together, such as 'animal', 'athlete', or 'author' would also be an exciting thing to do. Another possibility if the grouping feature gets implemented, is to match entities up against related entities and for example show all authors related to a publisher. In general, it is safe to assume that being able to better represent the connections and relations that are contained in Wikipedia would greatly enhance the value of the system. This could be achieved either by using the Wikipedia Link Graph [6], or by tapping more heavily into the semantics provided by for example DBpedia [1].

Chapter 12

Conclusion

The motivation for this project was to implement a system that allows the user to browse page view statistics and edit counts for Wikipedia entities. In order to be able to achieve the requirements presented in Chapter 1, we needed suitable platforms to analyse and store the vast amounts of data. Furthermore, the page view statistics are a perfect source for the detection of bursts and events for entities, and we wanted to give the user the opportunity to browse these as well. Finally, we wanted to add additional depth to the system by improving the accuracy when searching for entities, and by showing the user related entities.

One of the focal points in this project was to be able to structure the data into a scalable environment with a fast lookup time. This was realised by designing a HBase database scheme. To calculate and insert all the data into desired tables we had to create seven different jobs, where six of them utilise MapReduce. These jobs add page views, edit counts, bursts and events, Jaccard similarities, related entities and redirects into the database. The burst detection in this system is implemented using a sliding window. We implemented two different methods to determine if an entity was bursting, namely 'Z-scores' and 'Modified Z-scores'. We chose various thresholds to ensure that we only detect the more extreme outliers. Event detection is carried out by analysing the edit count. Lastly, we created a web application that gives the user a comprehensive view of entities and bursts and events.

In order to test our system, especially the burst and event detection, we chose a case study in the form of football transfers. More specifically, all football transfers in August 2011. Our hypothesis was that we would be able to detect most significant transfers in the form of events, and that we would even be able to predict transfers before they happen, due to additional page views being generated through publicity in newspapers and social media. As we conducted our experiments, we felt that some of the bursts should have been classified as events. The reason for this is the usage of a naïve threshold for the event detection, which favours the popular entities. Despite this, our burst and event detection is able to find roughly 4 out of 5 significant transfers, with only Wikipedia statistics as a source. Beyond that, we are even able to predict that 7 to 9 out of 10 incidents of entities bursting due to rumours will end up in a transfer, depending on the method we use.

A final remark about our experiments is that we conducted them on a small dataset. As a consequence of this, we were not able to gather a lot of meaningful results from the Jaccard

similarities of entities bursting patterns. However, we anticipate that this functionality would generate better results if we run our system on a bigger dataset.

Lastly, we want to state that the system fulfils all of the requirements we presented in Chapter 1. Even though we use a small dataset, our system performs admirably, and we receive even better results than we anticipated. This project has given us great insight into the field of Big Data, through the processing and storing of the vast amount of data. It has greatly enhanced our knowledge, and given us further insight into the possibilities and limitations offered by the technologies we have been utilising.

Bibliography

- [1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference, ISWC/ASWC 2007*, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.
- [2] Krisztian Balog, Gilad Mishne, and Maarten de Rijke. Why Are They Excited?: Identifying and Explaining Spikes in Blog Mood Levels. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, EACL '06*, pages 207–210, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.
- [3] Rick Cattell. Scalable SQL and NoSQL Data Stores. *SIGMOD Record*, 39(4):12–27, May 2011.
- [4] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7, OSDI '06*, pages 15–15, Berkeley, CA, USA, 2006. USENIX Association.
- [5] Prabhakar Raghavan Christopher D. Manning and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [6] Marek Ciglan and Kjetil Nørkvåg. WikiPop: Personalized Event Detection System based on Wikipedia Page View Statistics. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10*, pages 1931–1932, New York, NY, USA, 2010. ACM.
- [7] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):107–113, January 2008.
- [8] Oliver Ferschke, Torsten Zesch, and Iryna Gurevych. Wikipedia Revision Toolkit: Efficiently Accessing Wikipedia’s Edit History. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. System Demonstrations*, pages 97–102, Portland, OR, USA, Jun 2011.
- [9] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Huan Liu, and Philip S. Yu. Time-Dependent Event Hierarchy Construction. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '07*, pages 300–309, New York, NY, USA, 2007. ACM.

- [10] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. DEMON: Mining and Monitoring Evolving Data. *IEEE Transactions on Knowledge and Data Engineering*, 13 (1):50–63, January 2001.
- [11] Johannes Gehrke, Flip Korn, and Divesh Srivastava. On Computing Correlated Aggregates Over Continual Data Streams. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, pages 13–24, New York, NY, USA, 2001. ACM.
- [12] Lars George. *HBase: The Definitive Guide*. O'Reilly Media, 2011.
- [13] Mihai Georgescu, Nattiya Kanhabua, Daniel Krause, Wolfgang Nejdl, and Stefan Siersdorfer. Extracting Event-Related Information from Article Updates in Wikipedia. In *Proceedings of the 35th European Conference on Advances in Information Retrieval*, ECIR'13, pages 254–266, Berlin, Heidelberg, 2013. Springer-Verlag.
- [14] Mihai Georgescu, Dang Duc Pham, Nattiya Kanhabua, Sergej Zerr, Stefan Siersdorfer, and Wolfgang Nejdl. Temporal Summarization of Event-Related Updates in Wikipedia. In *Proceedings of the 22nd International Conference Companion on World Wide Web*, WWW '13 Companion, pages 281–284, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.
- [15] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web*. Chapman and Hall/CRC, 2010.
- [16] Boris Iglewicz and David C. Hoaglin. *How to Detect and Handle Outliers*. ASQC Basic References in Quality Control. ASQC Quality Press, 1993.
- [17] Marcel Karnstedt, Daniel Klan, Christian Pölitz, Kai-Uwe Sattler, and Conny Franke. Adaptive Burst Detection in a Stream Engine. In *Proceedings of the 2009 ACM symposium on Applied Computing*, SAC '09, pages 1511–1515, New York, NY, USA, 2009. ACM.
- [18] Jon Kleinberg. Bursty and Hierarchical Structure in Streams. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 91–101, New York, NY, USA, 2002. ACM.
- [19] Vamshi Krishna Konishetty, K. Arun Kumar, Kaladhar Voruganti, and G. V. Prabhakara Rao. Implementation and Evaluation of Scalable Data Structure over HBase. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, ICACCI '12, pages 1010–1018, New York, NY, USA, 2012. ACM.
- [20] Chuck Lam. *Hadoop in Action*. Manning Publications Co., 2010.
- [21] Miles Osborne, Saša Petrovic, Richard McCreadie, Craig Macdonald, and Iadh Ounis. Bieber no more: First Story Detection using Twitter and Wikipedia. In *SIGIR 2012 Workshop on Time-aware Information Access*, TAIA '12, New York, NY, USA, 2012. ACM.
- [22] Nish Parikh and Neel Sundaresan. Scalable and Near Real-Time Burst Detection from eCommerce Queries. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 972–980, New York, NY, USA, 2008. ACM.

- [23] Maria-Hendrike Peetz, Edgar Meij, and Maarten de Rijke. OpenGeist: Insight in the Stream of Page Views on Wikipedia. In *SIGIR 2012 Workshop on Time-aware Information Access*, TAIA '12, New York, NY, USA, 2012. ACM.
- [24] Saša Petrović, Miles Osborne, and Victor Lavrenko. Streaming First Story Detection with Application to Twitter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 181–189, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.
- [25] Jaroslav Pokorný. NoSQL Databases: A Step to Database Scalability in Web Environment. In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, iiWAS '11, pages 278–283, New York, NY, USA, 2011. ACM.
- [26] Tilmann Rabl, Sergio Gómez-Villamor, Mohammad Sadoghi, Victor Muntés-Mulero, Hans-Arno Jacobsen, and Serge Mankovskii. Solving Big Data Challenges for Enterprise Application Performance Management. *Proceedings of the Very Large Data Bases Endowment*, 5(12):1724–1735, August 2012.
- [27] Dennis Shasha and Yunyue Zhu. *High Performance Discovery In Time Series: Techniques And Case Studies (Monographs in Computer Science)*. Springer Verlag, 2004.
- [28] Thomas Steiner, Seth van Hooland, and Ed Summers. MJ no more: Using Concurrent Wikipedia Edit Spikes with Social Network Plausability Checks for Breaking News Detection. In *2013 WWW Workshop on Real-Time Analysis and Mining of Social Streams*, RAMSS '13, pages 791–794, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.
- [29] Ramine Tinati, Thanassis Tiropanis, and Leslie Carr. An Approach for Using Wikipedia to Measure the Flow of Trends Across Countries. In *Proceedings of the 22nd international conference on World Wide Web companion*, WWW '13 Companion, pages 1373–1378, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.
- [30] Michail Vlachos, Christopher Meek, Zografoula Vagena, and Dimitrios Gunopulos. Identifying Similarities, Periodicities and Bursts for Online Search Queries. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 131–142, New York, NY, USA, 2004. ACM.
- [31] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, 2012.
- [32] Stewart Whiting and Omar Alonso. Hashtags as Milestones in Time. In *SIGIR 2012 Workshop on Time-aware Information Access*, TAIA '12, New York, NY, USA, 2012. ACM.
- [33] Stewart Whiting, Ke Zhou, Joemon Jose, Omar Alonso, and Teerapong Leelanupab. CrowdTiles: Presenting Crowd-based Information for Event-driven Information Needs. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2698–2700, New York, NY, USA, 2012. ACM.

- [34] Jeonghee Yi. Detecting Buzz from Time-Sequenced Document Streams. In *Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05) on e-Technology, e-Commerce and e-Service*, EEE '05, pages 347–352, Washington, DC, USA, 2005. IEEE Computer Society.
- [35] Xin Zhang and Dennis Shasha. Better Burst Detection. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 146–, Washington, DC, USA, 2006. IEEE Computer Society.
- [36] Wayne Xin Zhao, Rishan Chen, Kai Fan, Hongfei Yan, and Xiaoming Li. A Novel Burst-based Text Representation Model for Scalable Event Detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 43–47, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.
- [37] Wayne Xin Zhao, Baihan Shu, Jing Jiang, Yang Song, Hongfei Yan, and Xiaoming Li. Identifying Event-related Bursts via Social Media Activities. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1466–1477, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.
- [38] Yunyue Zhu and Dennis Shasha. StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 358–369. VLDB Endowment, 2002.
- [39] Yunyue Zhu and Dennis Shasha. Efficient Elastic Burst Detection in Data Streams. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 336–345, New York, NY, USA, 2003. ACM.

Appendices

Appendix A

List of Transfers

A.1 BBC List

This list is retrieved from the BBC web site¹

31 AUGUST

Ahmed Abdulla [West Ham - Swindon] Loan
Tom Adeyemi [Norwich - Oldham] Loan
Mikel Arteta [Everton - Arsenal] £10m
David Ball [Peterborough - Rochdale] Loan
Jermaine Beckford [Everton - Leicester] £3m
Mark Beevers [Sheffield Wednesday - MK Dons] Loan
Craig Bellamy [Manchester City - Liverpool] Undisclosed
Ahmed Benali [Manchester City - Rochdale] Loan
Yossi Benayoun [Chelsea - Arsenal] Loan
Nicklas Bendtner [Arsenal - Sunderland] Loan
David Bentley [Tottenham - West Ham] Loan
Federico Bessone [Leeds - Swansea] Free
Villyan Bijev [California Odyssey - Liverpool] Undisclosed
Villyan Bijev [Liverpool - Fortuna Dusseldorf] Loan
Daniel Bogdanovic [Sheffield United - Blackpool] Undisclosed
Jordan Brown [West Ham - Aldershot] Loan
Shane Byrne [Leicester - Bury] Loan
Joel Campbell [Arsenal - Lorient] Loan
Aiden Chippendale [Huddersfield - Inverness Caledonian Thistle] Loan
Giles Coke [Sheffield Wednesday - Bury] Loan
Joe Cole [Liverpool - Lille] Loan
Peter Crouch [Tottenham - Stoke] £12m
Paul Currie [Berwick - Hamilton] Undisclosed
Scott Dann [Birmingham - Blackburn] Undisclosed
Ulises Davila [Chelsea - Vitesse Arnhem] Loan
David Davis [Wolves - Inverness Caledonian Thistle] Loan

¹<http://www.bbc.co.uk/sport/0/football/14365319>

Jack Deaman [unattached - Birmingham]
Guy Demel [Hamburg - West Ham] Undisclosed
Jamie Devitt [Hull - Bradford] Loan
Royston Drenthe [Real Madrid - Everton] Loan
Shane Duffy [Everton - Scunthorpe] Loan
Matt Duke [unattached - Bradford]
Badr El Kaddouri [Dynamo Kiev - Celtic] Loan
Wade Elliott [Burnley - Birmingham] Undisclosed
Liam Feeney [Bournemouth - Millwall] Undisclosed
Anton Ferdinand [Sunderland - QPR] Undisclosed
Jonathan Franks [Middlesbrough - Oxford] Loan
Joe Garner [Nottingham Forest - Watford] Undisclosed
Max Gradel [Leeds - St Etienne] Undisclosed
Jamie Griffiths [Ipswich - Plymouth] Loan
Zdenek Grygera [unattached - Fulham]
John Guidetti [Manchester City - Feyenoord] Loan
Rafik Halliche [Fulham - Swansea] Loan
Owen Hargreaves [unattached - Manchester City]
Shaun Harrad [Northampton - Bury] Undisclosed
Jos Hooiveld [Celtic - Southampton] Loan
James Hurst [West Brom - Blackpool] Loan
Alan Hutton [Tottenham - Aston Villa] Undisclosed
Pablo Ibanez [West Brom - Birmingham] Undisclosed
Jermaine Jenas [Tottenham - Aston Villa] Loan
Cameron Jerome [Birmingham - Stoke] Undisclosed
Gael Kakuta [Chelsea - Bolton] Loan
Simon King [Gillingham - Plymouth] Loan
Shefki Kuqi [unattached - Oldham]
Henri Lansbury [Arsenal - West Ham] Loan
Andy Little [Rangers - Port Vale] Loan
Ryan Lowe [Bury - Sheffield Wednesday] Undisclosed
Hector Mackie [unattached - Hayes & Yeading]
Shaun Maloney [Celtic - Wigan] £1m
Cody McDonald [Norwich - Coventry] Undisclosed
Ryan McGivern [Manchester City - Bristol City] Loan
Anthony McNamee [Norwich - MK Dons] Free
Raul Meireles [Liverpool - Chelsea] Undisclosed
Per Mertesacker [Werder Bremen - Arsenal] Undisclosed
Damian Mozika [Bury - Scunthorpe] Undisclosed
David Ngog [Liverpool - Bolton] Undisclosed
Dany N'Guessan [Leicester - Millwall] Undisclosed
Dean Overson [unattached - Bradford]
Wilson Palacios [Tottenham - Stoke] Undisclosed
Scott Parker [West Ham - Tottenham] £5m
Christian Poulsen [Liverpool - Evian] Undisclosed
Alex Revell [Leyton Orient - Rotherham] Undisclosed

Bryan Ruiz [Twente - Fulham] £10.6m
Orlando Sa [unattached - Fulham]
Andre Santos [Fenerbahce - Arsenal] £6.2m
Jay Simpson [Hull - Millwall] Loan
Emile Sinclair [Macclesfield - Peterborough] Undisclosed
Darnel Situ [Lens - Swansea] £250,000
Michael Smith [Darlington - Charlton] Undisclosed
Junior Stanislas [West Ham - Burnley] Undisclosed
Denis Stracqualursi [Tigre - Everton] Loan
Enda Stevens [Shamrock Rovers - Aston Villa] Undisclosed²
Gilles Sunu [Arsenal - Lorient] Undisclosed
Ben Swallow [Bristol Rovers - Bath City] Loan
Callum Tapping [Tottenham - Hearts] Undisclosed
Michael Timlin [Swindon - Southend] Loan
Ben Turner [Coventry - Cardiff] Undisclosed
Patrick Van Aanholt [Chelsea - Wigan] Loan
Martyn Waghorn [Leicester - Hull] Loan
Jed Wallace [Lewes - Portsmouth] Undisclosed
Shaun Wright-Phillips [Manchester City - QPR] Undisclosed
Yakubu [Everton - Blackburn] Undisclosed

30 AUGUST

Tom Bender [Colchester - Accrington] Loan
Sebastian Coates [Nacional - Liverpool] Undisclosed
Albert Crusat [Almeria - Wigan] Undisclosed
Papa Bouba Diop [unattached - West Ham]
Rob Elliot [Charlton - Newcastle] Undisclosed
Steven Howarth [Motherwell - Alloa] Loan
David Kasnik [Olimpija Ljubljana - Sheffield Wednesday] Loan
Daniel Kearns [Dundalk - Peterborough] Undisclosed
Keanu Marsh-Brown [Fulham - Dundee United] Loan
Ross McKinnon [Motherwell - Dumbarton] Loan
Kyel Reid [unattached - Bradford]
Davide Santon [Inter Milan- Newcastle] Undisclosed
Armand Traore [Arsenal - QPR] Undisclosed
Gerhard Tremmel [unattached - Swansea]
Josh Walker [Watford - Stevenage] Loan

29 AUGUST

Leon Cort [Burnley - Charlton] Loan
Kieran Djilali [unattached - AFC Wimbledon]
Jean Makoun [Aston Villa - Olympiakos] Loan
Brian Montenegro [Club Deportivo Maldonado - West Ham] Loan

²Deal to go through January 2012

Roque Santa Cruz [Manchester City - Real Betis] Loan

28 AUGUST

Brett Williams [Reading - Rotherham] Loan

27 AUGUST

Ulises Davila [Chivas - Chelsea] Undisclosed
Jonathan Hogg [Aston Villa - Watford] Undisclosed
Emiliano Insua [Liverpool - Sporting Lisbon] Undisclosed
Adam Le Fondre [Rotherham - Reading] Undisclosed
Luke Young [Aston Villa - QPR] Undisclosed

26 AUGUST

Sam Baldock [MK Dons - West Ham] Undisclosed
Thomas Barkhuizen [Blackpool - Hereford] Loan
Joey Barton [Newcastle - QPR] Free
Billy Bodin [Swindon - Torquay] Loan
David Crawford [Hibernian - Brechin] Loan
Will Evans [Swindon - Hereford] Loan
Lee Johnson [Bristol City - Chesterfield] Loan
Billy Kee [Torquay - Burton] Undisclosed
Charlie MacDonald [Brentford - MK Dons] Undisclosed
Shaun MacDonald [Swansea - Bournemouth] £80,000
Scott Smith [Hibernian - Brechin] Loan
Ibrahima Sonko [unattached - Ipswich]
Andreas Weimann [Aston Villa - Watford] Loan
Craig Westcarr [Notts County - Chesterfield] Undisclosed

25 AUGUST

Emmanuel Adebayor [Man City - Tottenham] Loan
Matthew Barnes-Homer [Luton - Rochdale] Loan
Jimmy Bullard [unattached - Ipswich]
Ashley Eastham [Blackpool - Bury] Loan
Jonathan Grounds [Middlesbrough - Chesterfield] Loan
Gavin Mahon [unattached - Notts County]
Daryl Murphy [Celtic - Ipswich] Loan
Adam Thompson [Watford - Brentford] Loan

24 AUGUST

Lauri Dalla Valle [Fulham - Dundee United] Loan
Kaspars Gorkss [QPR - Reading] Undisclosed

Juan Mata [Valencia - Chelsea] £23.5m
Samir Nasri [Arsenal - Man City] £25m

23 AUGUST

Danny Drinkwater [Manchester United - Barnsley] Loan
Cameron Park [Middlesbrough - Barnsley] Loan
Lee Miller [Middlesbrough - Carlisle] Free
Guirane N'Daw [St Etienne - Birmingham City] Loan
Simon Vukcevic [Sporting Lisbon - Blackburn] Undisclosed

22 AUGUST

Sotirios Kyrgiakos [Liverpool - Wolfsburg] Undisclosed
Ryan Harley [Swansea - Brighton] Undisclosed
Lukas Magera [Politehnica Timisoara - Swindon] Undisclosed

19 AUGUST

Pim Balkestein [Brentford - Rochdale] Loan
Joel Campbell [Deportivo Saprissa - Arsenal] undisclosed
Leon Clarke [QPR - Swindon] Free
Nathan Clarke [Huddersfield - Oldham] Loan
Bradley Diallo [unattached - Oldham]
Michael Doughty [QPR - Crawley] Loan
Lander Gabilondo [unattached - Swindon]
Anthony Gardner [unattached - Crystal Palace]
Milan Lalkovic [Chelsea - Doncaster] Loan

18 AUGUST

Simon Clist [Oxford - Hereford] Loan
Faris Haroun [unattached - Middlesbrough]
Jimmy Keohane [unattached - Exeter]
Zavon Hines [West Ham - Burnley] Tribunal

17 AUGUST

Ben Gordon [Chelsea - Peterborough] Loan
Ben Marshall [Stoke - Sheffield Wednesday] Loan
Carlos Salcido [Fulham - Tigres] Loan

16 AUGUST

Daniel Ayala [Liverpool - Norwich] Undisclosed
Emmanuel Eboué [Arsenal - Galatasaray] £3m

Adam Smith [Tottenham - MK Dons] Loan
Carlos Vela [Arsenal - Real Sociedad] loan

15 AUGUST

Michail Antonio [Reading - Colchester] Loan
Cesc Fabregas [Arsenal - Barcelona] Undisclosed
Wes Fletcher [Burnley - Accrington] Loan
Exodus Geohaghon [unattached - Barnet]
Danny Ings [Bournemouth - Burnley] Undisclosed
Robbie Keane [Tottenham - LA Galaxy] Undisclosed
Andy Keogh [Wolves - Leeds] Loan
Ishmael Miller [West Brom - Nottingham Forest] £1.2m
Kudus Oyenuga [Tottenham - Bury] Loan

13 AUGUST

Benjani [unattached - Portsmouth]
Steven Smith [Norwich - Preston] Free

12 AUGUST

Keith Andrews [Blackburn - Ipswich] Loan
Julio Arca [unattached - Middlesbrough]
Will Atkinson [Hull - Plymouth] Loan
Nouha Dicko [unattached - Wigan]
Alan O'Brien [unattached - Yeovil]
Jose Enrique [Newcastle - Liverpool] Undisclosed

11 AUGUST

Michael Bryan [Watford - Bradford] Loan
Ryan Doble [Southampton - Bournemouth] Loan
Danny Fox [Burnley - Southampton] Undisclosed
Chris Lines [Bristol Rovers - Sheffield Wednesday] £50,000
Bruno Perone [unattached - QPR]
James Tavernier [Newcastle - Carlisle] Loan
Danny Uchechi [FC Dender - Sheffield Wednesday] Loan

A.2 Refined List

This list is manually edited by the authors.

31 AUGUST

Mikel Arteta [Everton - Arsenal] £10m
Jermaine Beckford [Everton - Leicester] £3m
Craig Bellamy [Manchester City - Liverpool] Undisclosed
Yossi Benayoun [Chelsea - Arsenal] Loan
Nicklas Bendtner [Arsenal - Sunderland] Loan
David Bentley [Tottenham - West Ham] Loan
Federico Bessone [Leeds - Swansea] Free
Joel Campbell [Arsenal - Lorient] Loan
Joe Cole [Liverpool - Lille] Loan
Peter Crouch [Tottenham - Stoke] £12m
Scott Dann [Birmingham - Blackburn] Undisclosed
Ulises Davila [Chelsea - Vitesse Arnhem] Loan
Guy Demel [Hamburg - West Ham] Undisclosed
Royston Drenthe [Real Madrid - Everton] Loan
Badr El Kaddouri [Dynamo Kiev - Celtic] Loan
Anton Ferdinand [Sunderland - QPR] Undisclosed
Zdenek Grygera [unattached - Fulham]
Owen Hargreaves [unattached - Manchester City]
Alan Hutton [Tottenham - Aston Villa] Undisclosed
Jermaine Jenas [Tottenham - Aston Villa] Loan
Cameron Jerome [Birmingham - Stoke] Undisclosed
Gael Kakuta [Chelsea - Bolton] Loan
Henri Lansbury [Arsenal - West Ham] Loan
Shaun Maloney [Celtic - Wigan] £1m
Cody McDonald [Norwich - Coventry] Undisclosed
Raul Meireles [Liverpool - Chelsea] Undisclosed
Per Mertesacker [Werder Bremen - Arsenal] Undisclosed
David Ngog [Liverpool - Bolton] Undisclosed
Wilson Palacios [Tottenham - Stoke] Undisclosed
Scott Parker [West Ham - Tottenham] £5m
Christian Poulsen [Liverpool - Evian] Undisclosed
Bryan Ruiz [Twente - Fulham] £10.6m
Andre Santos [Fenerbahce - Arsenal] £6.2m
Denis Stracqualursi [Tigre - Everton] Loan
Patrick Van Aanholt [Chelsea - Wigan] Loan
Shaun Wright-Phillips [Manchester City - QPR] Undisclosed
Yakubu [Everton - Blackburn] Undisclosed

30 AUGUST

Sebastian Coates [Nacional - Liverpool] Undisclosed
Papa Bouba Diop [unattached - West Ham]
Rob Elliot [Charlton - Newcastle] Undisclosed
Davide Santon [Inter Milan- Newcastle] Undisclosed
Armand Traore [Arsenal - QPR] Undisclosed

29 AUGUST

Jean Makoun [Aston Villa - Olympiakos] Loan
Roque Santa Cruz [Manchester City - Real Betis] Loan

27 AUGUST

Ulises Davila [Chivas - Chelsea] Undisclosed
Emiliano Insua [Liverpool - Sporting Lisbon] Undisclosed

26 AUGUST

Joey Barton [Newcastle - QPR] Free

25 AUGUST

Emmanuel Adebayor [Man City - Tottenham] Loan

24 AUGUST

Juan Mata [Valencia - Chelsea] £23.5m
Samir Nasri [Arsenal - Man City] £25m

23 AUGUST

Simon Vukcevic [Sporting Lisbon - Blackburn] Undisclosed

22 AUGUST

Sotirios Kyrgiakos [Liverpool - Wolfsburg] Undisclosed

19 AUGUST

Joel Campbell [Deportivo Saprissa - Arsenal] undisclosed

16 AUGUST

Emmanuel Eboue [Arsenal - Galatasaray] £3m

Carlos Vela [Arsenal - Real Sociedad] loan

15 AUGUST

Cesc Fabregas [Arsenal - Barcelona] Undisclosed

Robbie Keane [Tottenham - LA Galaxy] Undisclosed

Appendix B

Google Search

In this chapter the Google searches conducted on the bursts and events described in Chapter 9 is displayed.

B.1 Bursts

Q

Web Images Maps Shopping More ▾ Search tools

About 695 results (0.38 seconds)

[Mikel John Obi - Wikipedia, the free encyclopedia](#)
en.wikipedia.org/wiki/Mikel_John_Obi ▾
 John Michael Nchekwube Obinna (born 22 April 1987), commonly known as **Mikel John Obi**, John Obi Mikel or John Mikel Obi, is a Nigerian footballer, who ...

[Mikel John Obi, Chelsea - Football Stats | Sky Sports](#)
www.skysports.com/football/player/0,,11668_317560,00.html ▾
 Chelsea Football Player **Mikel John Obi** - born APR 22, 1987 in Jos, Nigeria. Find player details, stats, transfer history and the latest news.

[Chelsea footballer's father kidnapped Nigeria](#)
www.smh.com.au > [Sport](#) > [Football](#) ▾
 Aug 16, 2011 – The father of Chelsea and Nigeria soccer star **John Obi Mikel** has been abducted in his homeland, but no ransom demand has been made, the ...

[Mikel John Obi appeals to nation to : London: The News](#)
www.londonthenews.com/.../20110815/.../Mikel-John-Obi-appeals-to-na... ▾
 Aug 15, 2011 – 'To whoever has got my dad, please just let him go' • **Obi Sr** did not return from work in Jos, Nigeria on Friday The ... London: The News on ...

[Mikel pleads for dad's release | Sport24 - News24](#)
m.news24.com > [Sport Home](#) > [Soccer](#) > [English Premiership](#) ▾
 Aug 15, 2011 – Chelsea footballer **John Obi Mikel** has begged for his father's kidnappers to release him after he went missing last week.

[Secuestraron al padre de John Obi Mikel - Fútbol ... - Iberoamerica.net](#)
www.iberoamerica.net > ... > eluniversal.com ▾ [Translate this page](#)
 Aug 15, 2011 – RT @EIUniversal: Secuestraron al padre de **John Obi Mikel** - http://t.co/SIOKEm7 · 15 ago 2011 15:31 · luis_kor. Lamentable Noticia RT ...

[Secuestraron al padre de John Obi Mikel - Fútbol ... - Iberoamerica.net](#)
www.iberoamerica.net > ... > eluniversal.com ▾ [Translate this page](#)
 Aug 15, 2011 – Menciones en. Este artículo se mencionó los siguientes días: Etiquetas más usadas para marcar este artículo: globovision. Secuestraron al ...

[Chelsea star's dad kidnapped | Sport24](#)
www.sport24.co.za/Soccer/.../Chelsea-stars-dad-kidnapped-20110815 ▾
 Aug 15, 2011 – The father of Chelsea's star midfielder, **John Obi Mikel**, has been abducted in his homeland of Nigeria, according to reports.

Figure B.1: Google Search - Mikel John Obi 15 August 2011

20110815 kolo touré

Web Images Maps Shopping More Search tools

About 214 results (0.48 seconds)

[Kolo Touré - Wikipedia, the free encyclopedia](#)
en.wikipedia.org/wiki/Kolo_Touré

Kolo Habib **Touré** (born 19 March 1981) is an Ivorian footballer who plays for Manchester City and the Côte d'Ivoire national football team as a central defender.

[Yaya Touré - Wikipedia, the free encyclopedia](#)
en.wikipedia.org/wiki/Yaya_Touré

The move teamed **Touré** up with his elder brother **Kolo**, who signed for City in July 2009 from Arsenal. On 28 July, **Touré** made his debut for City in a pre-season ...

[20110815 EPL Manchester City News News Updates - Kishys.com](#)
m.kishys.com/headlines-20110815-EPL_Manchester_City_News.html

Aug 15, 2011 – **20110815** EPL Manchester City News News Updates. ... **Kolo Toure** training with Man City after ban · Martin Samuel: Fail your Italian test and ...

[More News - ESPN SoccerNet](#)
soccer.net.espn.go.com/news/more?section=facup...20110815...

Aug 15, 2011 – City defender Toure returns to training (Aug 15, 2011 12:40 GMT)
Manchester City defender **Kolo Toure** has returned to training after the ...

[Banned Toure training again ahead of City's season opener - 7M sport](#)
www.7msports.net/news/newsdata/20110815/60929_2.shtml

Aug 15, 2011 – 聽聽Initially, **Toure** was told he was not allowed to play or train during that time. However, it is understood the defender has been taking part in ...

Figure B.2: Google Search - Kolo Touré 15 August 2011

B.2 Events

Q

Web
Images
Maps
Shopping
More ▾
Search tools

About 661 results (0.36 seconds)

[Bryan Ruiz - Wikipedia, the free encyclopedia](#)
en.wikipedia.org/wiki/Bryan_Ruiz ▾
Bryan Jafet Ruiz González (born 18 August 1985) is a Costa Rican footballer who plays for Fulham in the Premier League. He is a right sided attacking ...

[Bryan Ruiz - Player profile - transfermarkt.co.uk](#)
www.transfermarkt.co.uk/en/bryan-ruiz/profil/spieler_39642.html ▾
 11 **Bryan Ruiz**. Fulham FC, Premier ... The profile for **Bryan Ruiz**. **Bryan Ruiz** © Getty Images. Name in native country: Bryan Jafet Ruiz González. Date of birth ...

[As it happened: Transfer Deadline Day - RTÉ Sport](#)
www.rte.ie/sport/soccer/2011/0831/transferdaylive.html ▾
 Sep 1, 2011 – 2338 FC Twente striker **Bryan Ruiz**, who was a target for Newcastle United, has joined Fulham. The Cottagers were the first team in for the ...


[Fulham perto de anunciar **Bryan Ruiz** | record.xl.pt | | Portugal ...](#)
www.iberioamerica.net > ... > record.xl.pt ▾ Translate this page
 Aug 31, 2011 – 1, Fulham perto de anunciar **Bryan Ruiz** · http://www.record.xl.pt/Futebol/Internacional/interior.aspx?content_id=714801 ...

[Ruiz rumbo a Inglaterra | larepublica.net | | Costa Rica | 20110831 ...](#)
www.iberioamerica.net > ... > larepublica.net ▾ Translate this page
 Aug 31, 2011 – El Fulham se hace con un jugadorazo, **Bryan Ruiz** camino de ... RT @ La_Republica: **Bryan Ruiz** al fin dejará el Twente de Holanda para irse a ...

[Fox Sports y SkySports anuncian pase de **Bryan Ruiz** al Fulham ...](#)
www.iberioamerica.net > ... > nacion.com ▾ Translate this page
 Aug 31, 2011 – RT @nacion: Fox Sports y SkySports anuncian pase de **Bryan Ruiz** al ... **Bryan Ruiz** al Fulham por \$17.3 millones segun sky sport y fox sport ...

[More News - ESPN Soccernet - ESPN FC](#)
espnfc.com/news/more?date=20110831&ver=en ▾
 Aug 31, 2011 – Fulham complete deal for Twente's Ruiz (Aug 31, 2011 07:00 GMT)
 Fulham completed a £10.6 million move for FC Twente forward **Bryan Ruiz**.

Figure B.3: Google Search - Bryan Ruiz 31 August 2011

20110831 mikel arteta 

Web Images Maps Shopping More ▾ Search tools

About 371 results (0.35 seconds)


[Miguel Arteta - Wikipedia, the free encyclopedia](#)
en.wikipedia.org/wiki/Miguel_Arteta ▾
Miguel **Arteta** (born 1965) is a Puerto Rican director of film and television, known for his independent film *Chuck & Buck* (2000), for which he received the ...

[Socceroos revolution continues as Osieck faces some challenging ...](#)
www.smh.com.au > Sport > Football ▾
Sep 1, 2011 – 1 Sep Arsenal manager Arsene Wenger made a last-ditch swoop for Everton midfielder **Mikel Arteta** just hours after signing Fenerbahce ...

[Miguel Arteta - IMDb](#)
www.imdb.com/name/nm0037708/ ▾
Anne Heche and Miguel **Arteta** at event of Cedar Rapids Anne Heche, Miguel **Arteta**, Alia Shawkat, Ed Helms and Nancy Utley at event Miguel **Arteta** at event of ...

[Official Arsenal sign Mikel Arteta from Everton on : London: The News](#)
www.londonthenews.com/.../20110831/.../Official-Arsenal-sign-Mikel-A... ▾
Aug 31, 2011 – Spaniard completes last-minute deadline day switch to the Emirates following a six-year stay at Goodison Park, as Arsene Wenger ... London: ...

Figure B.4: Google Search - Mikel Arteta 31 August 2011

20110831 keisuke honda 

Web Images Maps Shopping More ▾ Search tools

About 816 results (0.21 seconds)

[Keisuke Honda - Wikipedia, the free encyclopedia](#)
en.wikipedia.org/wiki/Keisuke_Honda ▾
Keisuke Honda (本田 圭佑, Honda Keisuke, born 13 June 1986) is a Japanese footballer who plays for CSKA Moscow and the Japan national football team.
You visited this page on 5/9/13.

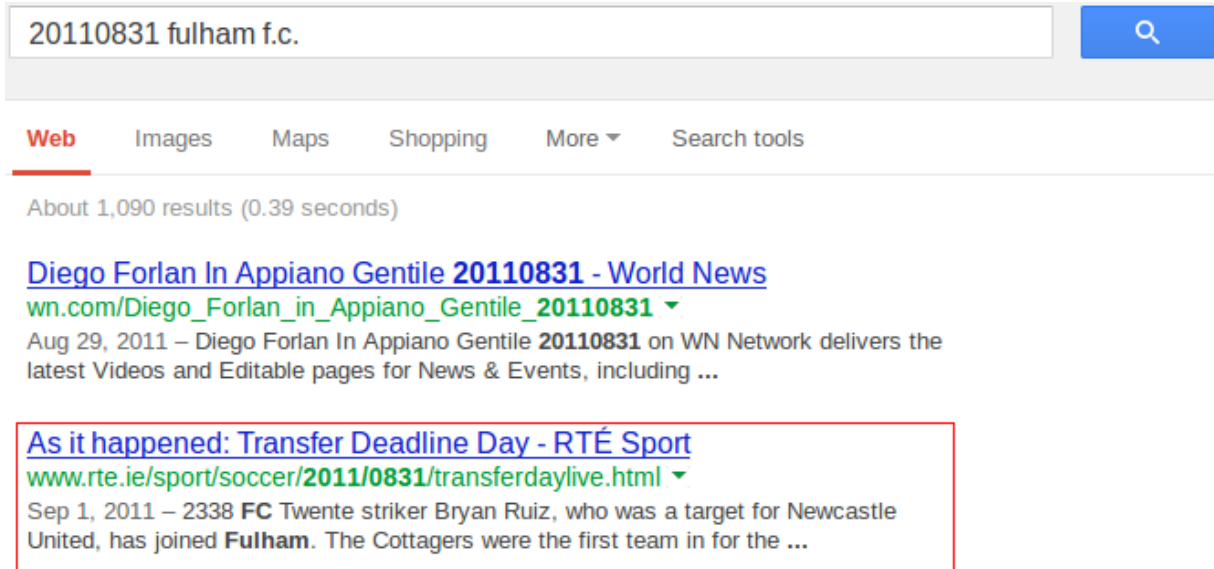
[Keisuke Honda - Player profile - transfermarkt.co.uk](#)
www.transfermarkt.co.uk/en/keisuke-honda/profil/spieler_66521.html ▾
7 **Keisuke Honda**. CSKA Moscow, Premier Liga ... Ravil Netfullin Alan Dzageev. The profile for **Keisuke Honda** ... News about **Keisuke Honda**. Coach Okada: ...

[As it happened: Transfer Deadline Day - RTÉ Sport](#)
www.rte.ie/sport/soccer/2011/0831/transferdaylive.html ▾
Sep 1, 2011 – 1145 The latest to bubble up from twitter and a few websites is that Japanese midfielder **Keisuke Honda** is going to move to Arsenal from CSKA ...

[Arsenal sign Santos & Mertesacker - RTÉ Sport](#)
www.rte.ie/sport/.../2011/0831/284192-arsenal_santosa_mertesacker/ ▾
Aug 31, 2011 – ... and Mikel Arteta along with Edin Hazard of Lille, CSKA Moscow's Japan international playmaker **Keisuke Honda** and Yann M'Vila at Rennes.
You visited this page on 5/9/13.

Figure B.5: Google Search - Keisuke Honda 31. August 2011

B.3 Related Bursts and Events



20110831 fulham f.c.

Web Images Maps Shopping More Search tools

About 1,090 results (0.39 seconds)

[Diego Forlan In Appiano Gentile 20110831 - World News](#)
wn.com/Diego_Forlan_in_Appiano_Gentile_20110831
Aug 29, 2011 – Diego Forlan In Appiano Gentile 20110831 on WN Network delivers the latest Videos and Editable pages for News & Events, including ...

[As it happened: Transfer Deadline Day - RTÉ Sport](#)
www.rte.ie/sport/soccer/2011/0831/transferdaylive.html
Sep 1, 2011 – 2338 **FC** Twente striker Bryan Ruiz, who was a target for Newcastle United, has joined **Fulham**. The Cottagers were the first team in for the ...

Figure B.6: Google Search - Fulham F.C. - 31 August 2011

Appendix C

DARPA Line Charts

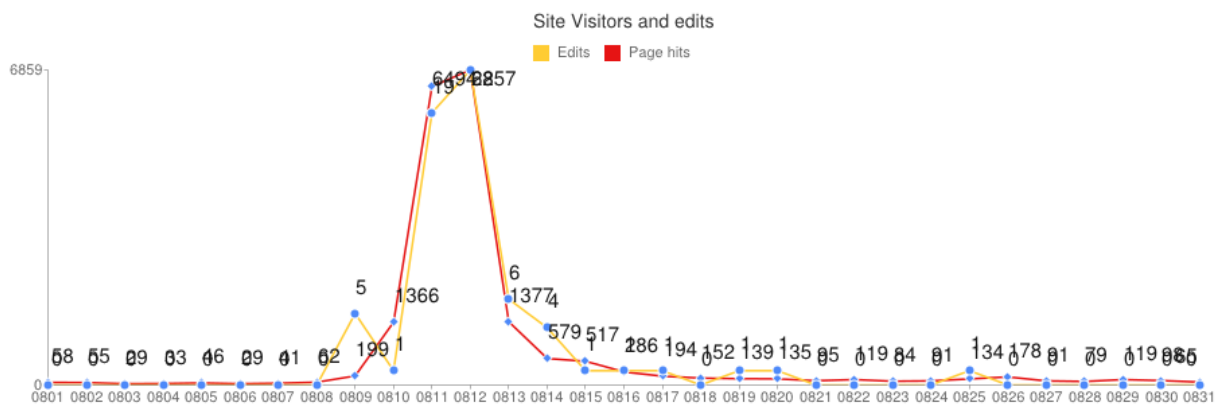


Figure C.1: Line Chart - DARPA's Falcon Project

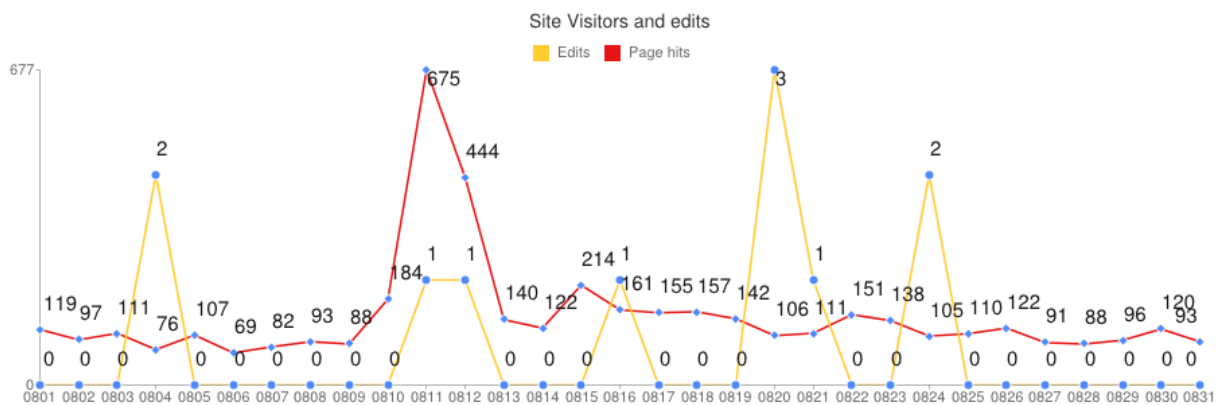


Figure C.2: Line Chart - DARPA

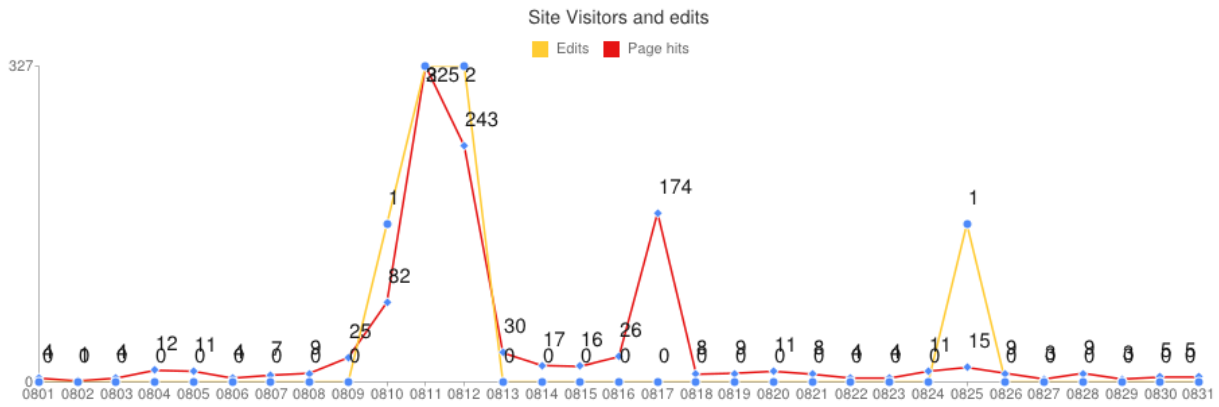


Figure C.3: Line Chart - Minotaur IV

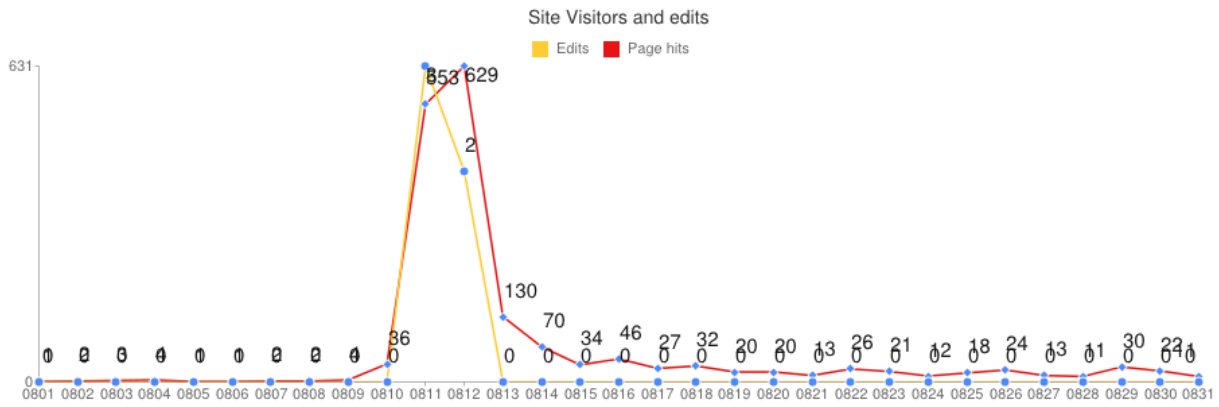


Figure C.4: Line Chart - HTV-2