Matchmood

Real-time comparison of betting markets and aggregate mood through sentiment analysis of social media.

Sean Bugeja
Department of Artificial Intelligence
University Of Malta
Msida, Malta
sean.bugeja.12@um.edu.mt

Matthew Joseph Zammit
Department of Artificial Intelligence
University Of Malta
Msida, Malta
matthew.zammit.09@um.edu.mt

ABSTRACT

The following paper shall present a system designed to monitor and extract sentiment information from twitter data streams related to sporting events. The system also tracks market information from betting companies in order to observe a plausible correlation between the sentiment analysis information from tweets and the reaction of the sports betting market. We obtained preliminary results from each data source and have visualised the sentiment reaction to specific points of interests during sporting events.

CCS Concepts

•Information systems → MapReduce languages; Sentiment analysis; *Column based storage;*

Keywords

MapReduce, Sentiment Analysis

1. INTRODUCTION

The astounding growth of social-media and micro-blogging platforms cannot be understated. These platforms have seen an exponential rise in their user-base year over year and, as such, has made them bountiful sources of information. The general public constantly use these platforms to share their opinion on anything and everything, and this has lead to a rise in the interest of mining this information effectively to obtain a detailed analysis of the general public opinion, or *mood*, with respect to a specific topic.
A similar statement could be made of the various betting platforms available on-line, which have experienced a constant growth in their popularity ever since the start of organised sports, taking over markets for other events such as elections and music competitions.
Highly available electronic markets and the advancements in Internet technology in recent years have contributed to the success and popularity of such platforms. One such platform which distinguishes itself from the rest is *Betfair*[1]. Betfair is not the typical bookmaker, where the odds set are driven by the profit needed for a betting company to keep operational and generate capital. Betfair is a sports exchange where it allows its users to trade bets on particular outcomes for a number of sporting events. Apart from backing outcomes to happen by betting on them, users can also lay events by taking on liability to pay the backers should the outcome they laid occurs. On such a platform, Betfair is a mere middle man facilitating the matching of opposite bets at the same prices. Meaning that the users themselves are the market makers by setting their own odds on their own terms, specifying their own return ratio based on their perception of an event occurring or not. *Betfair* then exposes an overall market value for an event based on these wagers, which should theoretically be representative of the users' belief (or *mood*) towards that event. Such an exchange system is called a prediction market and thus can be used to forecast the outcomes of upcoming events.
Herein lies the problem that this system is meant to tackle. The proposed system shall be tasked with the gathering, organisation, and sub-sequent cross-correlation of market value data and twitter[2] sentiment data for a specific event, obtained through both of the respective entities' API. Finally, the system shall present a graphical representation of the information obtained in an effort to better visualise the relationship, if any, of the two data sources.
The following paper is divided as follows, Section 2 shall present a brief overview of the state of the art within the fields related to this task as it pertains to this specific domain. Section 3 presents a thorough explanation of the methodology adopted throughout each module within the system as a whole. Section 4 lists the evaluation techniques used, as well as the results obtained from this evaluation. Finally, section 5 concludes with a brief summary of the work presented herein while also exploring possible future improvements to the system.

2. STATE OF THE ART

The process of sentiment analysis is a growing field within the scope of natural language processing with the over-arching goal of identifying the public mood or opinion on specific topic from textual data. This field has experiences a particular rise in popularity in recent years due to the boom of social-media[1] and the opportunities that this presents.
In carrying out sentiment analysis, two common approaches that are adopted are those based on lexicon resources and those based on some machine-learning algorithm. In the former, a lexicon dictionary is used, such as the NRC[4], which relates a set of commonly used words to negative, positive or

[1]Accessible through http://www.betfair.com

[2]A popular micro-blogging platform, accessible through http://www.twitter.com

neutral sentiments, as well as other emotions such as anger and joy. Each word within a piece of text is then cross references with this dictionary and values are obtained pertaining to each sentiment for a specific piece of text.
With the rise of social-media and micro-blogging platforms such as twitter, the amount of data that must be analysed has increased substantially and this has required the development of more sophisticated and distributed data storage systems which can cope with this amount of data.
Once such data store is the Apache Cassandra [3], which is a column-based data store. This different approach makes allows for astoundingly fast read and write speeds while also making it ideal for many time-series data, such as a stream of *tweets*.
The MapReduce programming model[2] presents itself as a powerful tool when used together with large data-stores such as Cassandra[3]. This model aims facilitate the use of parallel systems, with optimal performance, by reducing network bandwidth overhead within a cluster while also providing fault-tolerance across computations, all while abstracting the intricate details of the parallelization. The model allows for easily implementing parallel functions that can aggregate attributes across large data-stores grouped by some particular key.
Furthermore, the Apache Spark[6] program is one that builds on top of this MapReduce model while at the same time allowing operations such as `joins` to occur on data-stores such as Cassandra[3], which would otherwise be impossible. A spark cluster comprises of a master node which sends jobs to the respective slave nodes within the cluster, then finally produces the result of it's job carried out across the entire data-store.
All of these tools make the process of handling the ever-growing amounts of data present in various fields, such as social media, more effective and efficient. The work presented in this paper is loosely based on similar work carried out by [5] who tracked the sentiment of American sports fans during the soccer world cup. We build on top of this work by further collecting betting market data and examining the correlation between the two.

3. METHODOLOGY

The following section shall detail the methodology adopted throughout the design phase of the system, leading up to its evaluation, which shall be covered in the sub-sequent section. This section will provide a comprehensive report on the design choices made when building the system architecture, outline all problems encountered during this process as well as the sub-sequent alterations made to the system in an effort to overcome these problems and illustrate the connection of each design choice with the respective goal it aims to achieve. This information is organised in sections, each representing the distinct, principle, phases of the system's design.

3.1 Stream Interface

The initial step in designing the system was to interface it with the relevant APIs provided by *Twitter* and *Betfair*. For the purposes of the system, a constant stream of data is required from both sources so as to enable to continuous correlation between the two. For such a task, twitter provides a dedicated streaming API which provides a constant stream of live *tweets*, which can be further filtered through a specified *filter string*. Naturally, since twitter is a general social media platform, it contains no knowledge, or means of acquiring knowledge, about current sporting events. This information would have to be extracted from the *Betfair* data source.
This process would proceed as follows: initially, the system is provided with a specific competition to follow, such as the *'Barclay's Premier League'*, the system subsequently queries the *Betfair* API for current events taking place within this competition which contain the three primary event outcomes of any sporting match, namely: a win for the home team, a win for the away team or the game ties. Using this approach the system obtains a list of all current events for which a market is currently open. A particular complication encountered at this point is that for many events involving the same teams, the respective markets for games involving those teams would be open simultaneously. So as to overcome this, a further limitation is set at this point to limit the number of events extracted from a competition to a two-day time-frame. Otherwise, tweets directed at a particular event may be incorrectly set to the wrong match tag by the system. So apart from the constraints set in the twitter stream listener; to be discussed in the next sub section, the assumption that users will be tweeting about the most recent events is made. From the resultant events, the system is able to extract the team names as well as a unique identifier used by *Betfair* for each particular team.
From this information the system must construct the respective filter that would only allow *tweets* relevant to particular matches of interest. An automated system would be possible for such a task, given the official team names provided by the *Betfair* API, however this proves to be far more complex than one might think due to the diverse approaches different leagues take towards structuring their presence on social media. This, coupled with frequent, small yet breaking, inconsistencies between the official team name provided by *Betfair* and those used on *twitter*, led to the decision of manually building a dictionary of teams, each identified by its unique *Betfair* ID, and containing the teams' respective *twitter* handles (eg. `@ChelseFC`) and a three-letter abbreviation (eg. `CFC`) commonly used to identify a specific event by concatenating this into a hashtag with the opposing team's own three-letter abbreviation. Other hashtags found on the official twitter profiles of the chosen football clubs, such as team nick names, other abbreviations than the tri-character one, that refer to their team were also retrieved. This allows for the system gather more data from the stream. Thus, more information could be extracted from such tweets that would otherwise not be retrieved, as users are not bound to choose particular references to the same real world entities and thus may be inconsistent in the tags and handles they choose to direct their message to. From the dictionary built, the twitter steam can be effectively filtered for data pertaining only to specific sporting events, and the teams involved in those events, at the cost of having to manually update this dictionary with the relevant information to include more teams from other competitions.

3.2 Data Pre-Processing

Before storing the acquired data, some pre-processing is carried out to facilitate the extraction of information at later stages of the system. This is particularly true for the *twitter* stream which contains various artifacts common to social

media platforms. These artifacts would otherwise hinder the sentiment analysis process and must therefore be dealt with accordingly.
For each tweet, the following approach is adopted: first, the tweet is checked for any reference of a team through their twitter handle which it knows about; for each such mention, the official name of the team is substituted. Subsequently, the following artifacts are removed from each tweet: URLs, `RT` markers, carriage returns, line feeds, and tabs, user mentions, hashtags not containing any letters and any words containing symbols other than those appearing in the Latin alphabet (with exceptions for punctuation marks). Essentially the system aims to reduce all tweets to a series of legible words within the English dictionary.
From this resultant sentence, the system extracts a value for the sentiment (ranging from [-1,1]) and subjectivity (ranging from [0,1]) through the use of the TextBlob library for python. On top of this, all the words appearing within the tweet are cross-referenced with the NRC-English sentiment dictionary[4], for their respective sentiment contribution towards each of the following: Anger, Anticipation, Disgust, Fear, Joy, Negative, Positive, Sadness, Surprise. The values for each of these sentiments are accumulated within each tweet and normalised by the number of words appearing in the NRC Dictionary[4].
A setback encountered here was the particular encoding of tweets, having the tweet encoded in anything but ASCII was producing run-time exceptions, so as to mitigate this, the tweets were encoded in ASCII, and any UTF characters were ignored. When dealing with English tweets, this is not a serious problem, however utilising UTF would allow further sentiment analysis be carried out using *Emojis*, which are increasingly popular on social media. For both streams, the `MatchTime` is recorded, which is calculated as the amount of minutes from the current time to the start of the event (negative up until the start of the event).

3.3 Data Storage

Following the pre-processing phase, the data is stored in column-based format using Apache Cassandra[3] across two different tables, for the market values obtained from *Betfair*, and tweet data obtained from *Twitter*, which are themselves hosted on an Amazon Web Services (AWS) cluster comprising of four nodes in total, with a shared hdfs, three of which host the Cassandra[3] cluster (shown in figure 3). The cluster was setup manually as the approach that best fit the challenge at hand, required a very specific set of requirements to allow for the solution stack described above to be setup accordingly. The Cassandra tables and the respective attributes which are stored within them are illustrated in figures 1 and 2 respectively.
Within these tables, the `tag` attribute is used as a primary partition key. This is imperative since the system will most often be serving information related to a specific event which it identifies through this tag. Furthermore, `match_time` is used as a clustering key since this is a primary point of reference when carrying out all subsequent analysis, the `match_time` represents the resolution of the system. Finally, a `time_uuid` had to also be specified as a clustering key, allowing for multiple entries to be stored for the same `match_time`.
It is also important to note the `entity` attribute present in the *twitter* data-store, which is a character representation

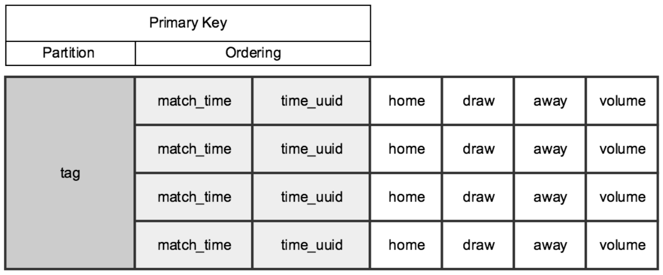

Figure 1: Betfair Column Datastore

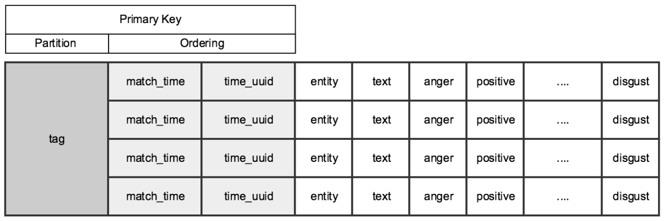

Figure 2: Twitter Column Datastore

of any user `handle` or `hashtag` within a tweet (prior to it being processed); if a tweet contains multiple *entities*, then it is stored once for each entity present.

3.4 Aggregation Techniques

The *raw* data stored within the aforementioned tables serves as the first level of storage, with which an Apache Spark[6] cluster interacts. This Spark[6] cluster is distributed across the four nodes, with three serving as slaves, while the fourth (which also houses the system core) serves as the master node, delegating the required Spark[6] jobs amongst the slaves as necessary. The Spark[6] cluster (illustrated in figure 4) allows the system to perform SQL type queries which would otherwise be impossible to carry out on the Cassandra[3] data-store by loading this data into its own Resilient Distributed Data (RDD) object.
Furthermore, MapReduce jobs are implemented within the Spark[6] jobs themselves (written in Scala) which allow for the distributed aggregation of *raw* data to produce four new Cassandra[3] tables containing the end-results. In total, three distinct Spark[6] jobs are defined, the details of which are covered hereunder.

3.4.1 Getting Distinct Match Tags

This is the simplest of the three jobs and serves the purpose of obtaining all distinct `match_tags` for the day, essentially providing a list of events which the system shall be or is currently monitoring.

3.4.2 Performing Sentiment Analysis

The second job handles the aggregation of data found within the *twitter* data-store, and outputs the result to two separate tables. The tables themselves are similar in all regards but one, namely the `match_time`. This allows for one table to provide information pertaining to the sentiment at a specific `match_time`, while the other to provide overall sentiment values as an average from when the event started being tracked. For the former, values for each sentiment are

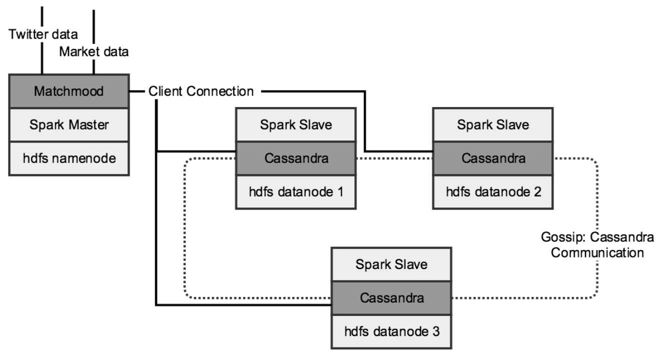

Figure 3: Cassandra Cluster Setup

aggregated as is, grouped by the specific `match_time`, while the latter aggregates all sentiment values for the event, then normalises this value by the amount of tweets thus far. In both cases the results are aggregated by the `match_tag` and subsequently the `entity`.

3.4.3 Performing Market Analysis

Finally, the third job is in charge of aggregating the *raw* market data. For the purpose of extracting multiple metrics from the `home` and `away` market values (the `draw` value is dropped at this time and is only stored for future improvement possibilities), these values are mapped three times over. This enables the Spark[6] job to extract the *min*, *max* and *avg* (by also aggregating a count value) values for both in a single run. These are joined with an ordered RDD with the same keys to ensure that all values are chronological. Finally from this data, the minimum, maximum, average and volume difference is extracted and stored by `match_tag`, `selection_type` (Home or Away) and `match_time`.

3.4.4 Spark Job Parameters

The aforementioned Spark[6] jobs are all executed against a string parameter which can be one of the following:

- ‘day’ : Instructs the Spark[6] head to execute the daily task, namely collecting all new match tags to follow.
- ‘latest’ : Carries out the sentiment analysis and market analysis tasks in near real time by extracting and calculating the required information from the past minute by the use of the time_stamp attribute.
- ‘rebuild’ : Is implemented primarily for debugging purposes to allow the recycling of old data which does not necessarily conform to the structure chosen for the final version of the system and allows for a batch processing approach making use of the match_time attribute stored with insert, instead of the time_stamp used for the near real time analysis.

3.5 Data Presentation

The final layer of the system is that which consumes the data provided through the aforementioned Spark[6] jobs, and illustrates it in a suitable format for the end user. This layer comprises of two primary modules, namely, a RESTful API and a graphical front-end which presents this information to the end user. A brief overview of these modules is provided herein.

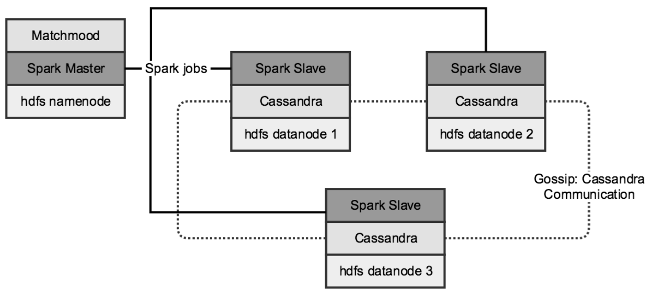

Figure 4: Spark Cluster Setup

3.5.1 RESTful API

A RESTful API shall be developed for the system using the Flask Python module, which exposes five endpoints which read from the end-result Cassandra[3] tables constantly being updated by the relevant Spark[6] jobs. A brief summary of the endpoints provided by the API is provided in table 1. All endpoints expose `GET` http methods, and return data in `JSON` format.

4. EVALUATION

In this section, the data collection process used for building and evaluating the data set is described in detail. Further more, results and conclusions obtained during the evaluation period are then discussed at length and shown in the form of charts obtained from the Graphical User Interface of the built system.

4.1 Data Collection

The Data collection process for the building and evaluation of the MatchMood project was an ongoing process. One of the main reasons this approach was taken was because there was no already defined data set that could be used. Thus, the data set was constructed in parallel with the development of the system. This kind of data collection process had its to challenges and obstacles. One of the main challenges encountered was that the critical information of a soccer like most other sporting events is played out moments before and through the duration of the event, which are both sparse and of a very short time slot of around 120 minutes. This meant that at any particular time the critical data was to be collected the current version of the system had to be functioning properly without any bugs that could hinder the process, which was not always the case. During this process the system was continuously being improved to better the data collection process, to allow for more data throughput, to optimise the system and the fixing of bugs that usually stopped the collection process. An other challenges encountered during this stage was getting familiar with the data modelling used by Cassandra. The data models were changed several times before arriving at the stage with the current version of the tables. This was an on going and incremental process that led to the project being at the current stable version, where the system is able to collect both market and twitter data instantaneously and perform near real time analytics such as aggregations, repopulate data sets with the results from the aggregations, expose a REST API to query the data and a GUI for visualising it.

URI Path	**Description**
/matches/	Retrieves a list of all distinct match tags in the system.
/matches/[tag]/market/	Retrieves market data for a specific match tag.
/matches/[tag]/entities/	Retrieves entities for a specified match tag.
/matches/[tag]/entities/[entity]	Retrieves sentiment overview for a specific entity under a specific match tag.
/matches/[tag]/entities/[entity]/minutes/	Retrieves per-minute sentiment data for an entity under a specific match tag.

Table 1: API Endpoint Description

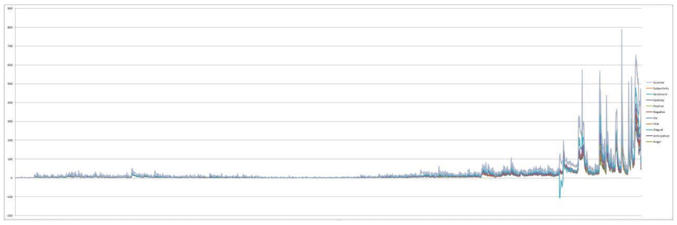

Figure 5: Sentiment Reaction for Crystal Palace

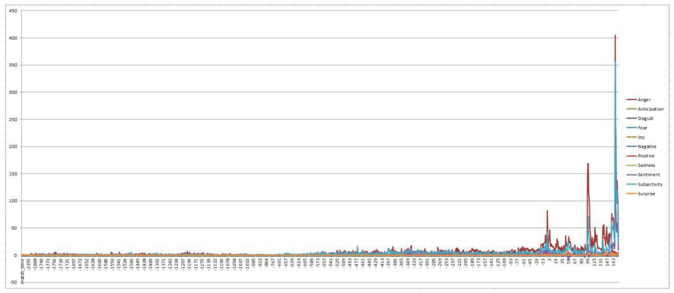

Figure 6: Sentiment Reaction for Manchester United

Taking aside these hindrances, around 35G worth of tweet and market data is distributed across the Cassandra cluster.

4.2 Results

The main aim of this research is the study of two very different ,yet similar domains, a prediction market and a social media platform. The study conducted in this research is to see whether there are any correlations between the two platforms for expressing one's opinion a particular event happening at real time and see whether any relationship exist between them. An other important study is to see how these two domains react upon arrival of new information.

5. CONCLUSIONS & FUTURE WORK

Social-media and prediction markets are two domains that in recent years have picked up momentum through the advancements of Internet technology and high availability systems. Although these two domains seem very different at first, they offer the same basic feature to their users; to express their opinion. In this research, the study of such relationships between these two domains is tackled by the use of big data solutions. In the literature study carried out it is found that such solution stacks are being used for such tasks and that similar research in this area is also being conducted. The problem at hand was tackled by the setting up of a Cassandra cluster made of 3 data nodes and spark slaves sitting on the same stack for the real time analytics, building clients for Betfair and Twitter data and also building a public API for interfacing. This solution is found to be both viable and practical as the jobs at hand are performed as requested.

For future development, as part of our original study, is to perform further analytics and build documents based on the time of match and the most n-grams used by the twitter user to try and predict the outcome of the match by what is being said. Also compare the value to that from betfair markets.

6. REFERENCES

[1] L. Barbosa and J. Feng. Robust Sentiment Detection on Twitter from Biased and Noisy Data. *Coling*, (August):36–44, 2010.

[2] J. Dean and S. Ghemawat. MapReduce: Simplied Data Processing on Large Clusters. *Proceedings of 6th Symposium on Operating Systems Design and Implementation*, pages 137–149, 2004.

[3] Laksham Avinash and Prashant Malik. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, pages 1–6, 2010.

[4] S. M. Mohammad and P. D. Turney. Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence*, 29(3):436–465, 2013.

[5] Y. Yu and X. Wang. World Cup 2014 in the Twitter World: A big data analysis of sentiments in U.S. sports fans' tweets. *Computers in Human Behavior*, 48(March):392–400, 2015.

[6] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster Computing with Working Sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, page 10, Berkeley, CA, USA, 2010. USENIX Association.