International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 02, February-2017

Automatic Notification Bot using Web Mining for Event Analysis

Mausam Yede, Anagha Akole, Chaitali Joshi, Sayali Pitekar
Dept. of Computer Engineering
Pune Institute of Computer Technology.
Pune, India

***Abstract*—In today's era, lot of important events happen all around the world. With the busy lives of an individual, it becomes difficult to keep track of all the events that they may be interested in. Online websites for different events enable users to get information, but at the same time users must know which website to visit, when to visit, and how to navigate through it to obtain relevant information. In this project, we suggest implementation of an online "bot" that will retrieve important information and updates about any trending event and automatically notify users about the highlights of the event in real time. The information acquired from the source will be mined using data mining algorithms and users will be notified about the associations, correlations, and other connections that may be deduced from the data. In this case, we present a system for a sport event, specifically cricket. The system will extract current scores for the ongoing match and using analysis based on historical data, try to predict the winning chances of a team. Our system can be extended for analysis of other types of events in future.**

Keywords*— *Data mining, Classification, Information search and retrieval, Information filtering.

I. INTRODUCTION

Today's world is fast and data is more important than ever. People are interested in the day to day activities happening all around the world. Because of internet, it is now very easy to access information about major events in real time and keep ourselves updated.

Events can be of many types, like sports, elections, competitions etc and generally have different sources of information. Hence, sometimes it becomes difficult to keep track of multiple events at the same time as the user may have to visit different websites for different events.

In this paper we propose a system that is capable of bringing information from one or more events to a common place for user's ease. The system is capable of extracting information from various real time sources of the event, aggregate it, analyze it, and deliver the obtained results in form of social media posts to the users. Hence, the user can access live information about more than one event on their social media account. Since social media sites like Facebook and Twitter already have the feature of sending notifications to the users, this information is literally just a look away.

To present a working system, we consider cricket match as the event to be analyzed. Specifically, One Day International (ODI) match played between the 9 ICC teams namely, India, Australia, South Africa, England, Sri Lanka, Pakistan, Bangladesh, West Indies, and New Zealand. More such events can be incorporated in the system for analysis and live information delivery, but that comes under future scope for this project.

This paper is divided into following sections: Section II describes the work done in the past related to our system. Section III presents design of the proposed system. Section IV includes the implementation details and algorithm used in the system. Section V gives details about the environment needed for system processing. Section VI includes the results obtained by our system. Section VII presents conclusion and Section VIII includes future work related to the system.

A. Brief introduction to Cricket

We present a basic overview of the game - Cricket. Cricket is the second most popular sport in the world with abundant followers across India, UK, Pakistan, Africa, Australia, etc. It is an outdoor game played on a field with 22-yard rectangular long pitch, between two teams each consisting of 11 players. There are 3 formats for a cricket game. They are Test, One Day International (ODI) and Twenty Over International (T20). Here, we propose a model mainly for prediction in ODI matches.

In an ODI match, each team takes turn batting while the other team bowls and fields. 50 overs are allotted to each team in which they try to score maximum runs. The duration is termed as an inning. So each ODI match consists of two inning

A cricket match starts with a toss. The team that wins the toss can choose whether to bat first or bowl first. This is a strategic decision that may influence the chances of victory.

Objective: Suppose team 1 wins the toss and decides to bat first, in this duration team 2 bowls, and the duration is known as inning 1. Team 1 has 50 overs to score as many runs as they can. Each over consists of 6 balls. The aim of team 2 is to minimize the runs scored by team 1 by delivering balls that are hard to score on or by exceptional fielding. Team 2 bowlers can remove the batsmen of team 1 by taking "wickets". Wickets can be take by the means of "run out", catching the ball before it touches the ground, by hitting the stumps behind the batsman with the ball or by LBW rule. Innings 1 comes to an end when team 1 loses all its wickets or finishes its allotted 50 overs, whichever happens first. The second inning is repeated for the other team with same rules.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 02, February-2017

If the total number of runs scored by team 1 in innings 1 is runsA, then the objective of team 2 is to score at least runsA+1 runs in innings 2 while the objective of team 1 is to take all the wickets or finish delivering all 50 overs before team 2 can reach this ”target”. If team 2 manages to score runsA+1 runs within the allotted 50 overs then it is deemed as winner of the match, otherwise team 1 wins. A third possibility is a tie between both the teams in which team 2 manages to score exactly runsA runs, i.e. both the teams score same amount of runs.

Scoring: Teams can score runs in two ways. One way of getting runs is to power-hit the ball outside the playing area. 4 runs are awarded if the ball touches the ground before going past the boundary of the playing field. These are known as ”fours” . If the ball lands directly outside the playing field, 6 runs are awarded. These are called as ”sixes”. ”Fours” and ”Sixes” yield more runs scored, but the batsmen have to take risks to hit them, which increases their chance of getting out. The other way of scoring runs is to hit the ball within the playing field and for the two currently playing batsmen to run and exchange their positions. In the mean time, the opponent fielders try to collect the ball to minimize the number of exchanges. Runs are given depending on the number of times the batsmen exchange their positions before the ball is returned to one of the positions. Generally, players score 1-3 runs at maximum per ball using this method. This way of scoring has a lower risk of the batsman losing wicket but yields a lower number of runs. We term these non-home runs. Runs are awarded to the batting team when the bowler commits a ”wide ball” or ”no ball” while delivering the ball. These are considered fouls on bowler’s behalf.

B. Event Analysis

Event analysis can be explained as performing various operations on live data for an event to find out interesting patterns in the event and hence giving predictions, correlation etc about the event.

In this paper we describe a system that is capable of relating the current cricket match situation to the past situations and predict the number of runs scored in an innings and hence predict the winner of the game by looking at the live data of the match.

We also provide other event information like current score, batsmen centuries/half centuries, fall of wickets, etc as and when they happen.

II. RELATED WORK

Several attempts have been made to create a model which can correctly predict the result of a cricket match. Some of these models are summarized here.

Sankarnarayanan, Sattar, and Lakshmanan [1] proposed a model for prediction of score and hence win/lose in an ODI (One Day International) cricket match. By using both supervised and unsupervised algorithms, this model takes into consideration a number of features from both instantaneous data and historical data, and gives separate methods for the prediction of home and non-home runs in the match. They have collected the data for all ODI matches that took place from January 2012 to June 2013 and used it for training the model.

The model takes into account both team aspects as well as individual calibers of batsman/bowlers in order to influence the prediction.

Tejinder Singh et al.[2] proposed a model that uses two methods to predict the score and winning chances in cricket (ODI). The first method predicts the score of the first innings based on the current run rate, venue of the match, number of wickets fallen and the batting team using Linear Regression Classifier.

The second method predicts the score of the second innings based on the target given to the batting team using Nave Bayes Classifier. The datasets are analyzed in Weka.

In a study presented by Fahad Munir, Md. Kamrul Hasan, Sakib Ahmed, Sultan Md. Quraish[3], the main goal is to develop a model to predict the outcome of a T20 cricket match while the game is in progress. Data of previous matches played between the team is used to design the model. Decision tree is used in this research along with Multiple Linear Regression in order to make a comparison of the results found. Decision tree algorithm is used to design a forecasting system by depending on the previous data of matches played between the teams. More emphasis is given on in-game attributes as the prediction will be when match is in progress.

III. SYSTEM DESIGN

The system is consists of three phases- Data extraction, Data processing and Data delivery.

A. Data extraction

Information about the ongoing event is extracted from the internet(web pages) during this phase. This is done by using a crawler or web API if one is available for the event. The data which is used for the event analysis is obtained in this manner. After extraction of the data, the data is filtered according to the needs of processing algorithm. Data extraction happens continuously for the duration of the event in order to periodically capture the live data of the event.

B. Data processing

Event analysis depends on the type of event and so does the processing. For example, in a cricket match the type of analysis can be predicting runs for an innings after looking at the current game play, predicting the runs for a particular batsman etc. This may vary for different events. In general, processing may include aggregation, prediction, clustering, classification, etc. Various algorithms can be used to perform this processing on the available data. The data for processing is obtained from the data extraction module after the extraction phase.

In this presented system, we notify users for following events in a live match:

1) Loss of wicket.
2) Century/half century for a batsman.
3) Current runs and wickets at intervals.
4) End of innings score prediction at end of every 5 overs.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 02, February-2017

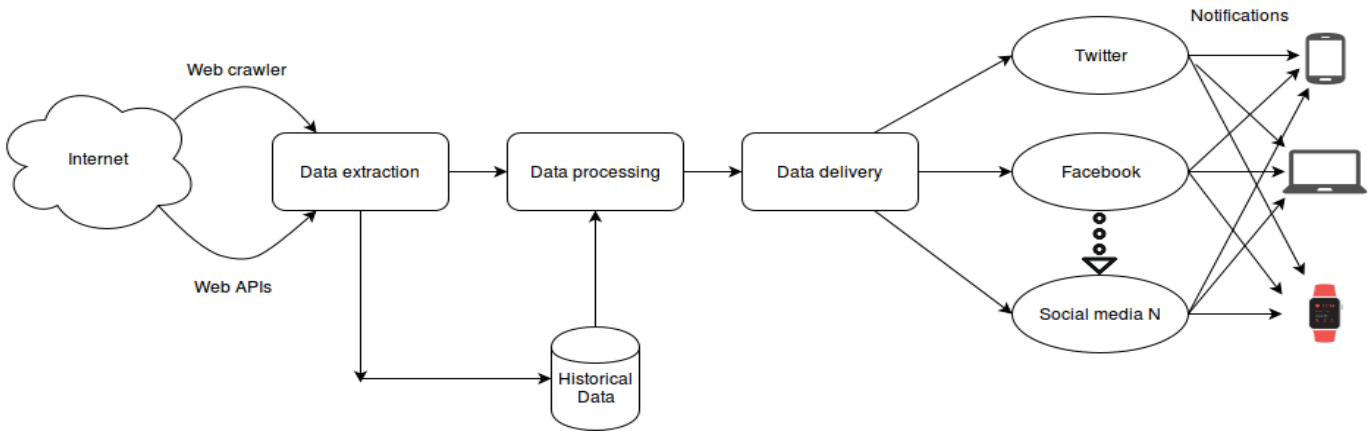
5) Win/lose prediction.
6) Current match status.
7) Segment wise prediction graph at the end of match.

C. Data Delivery

In this phase, the processed data is received by the delivery module and is then delivered to the user through notifications via social media sites like Facebook and Twitter. The processed results are posted on the social media sites automatically using RestAPIs available to the developers. This APIs are provided by the social media itself. Once posted, the users who use social media websites can receive the notifications on their Smartphone apps or on the internet browsers.

Figure 1 shows the system design architecture.

Fig. 1. System design



IV. IMPLEMENTATION

A. Extracting match data from the web

The initial stage of this system is to obtain live cricket match data from the websites that provide ball by ball match information in real time. For this purpose we have choice of online information sources like www.cricbuzz.com and www.espncricinfo.com.

In order to extract live match scores and other details from these websites, we use a web crawler made using python programming language. This crawler extracts information like current runs, wickets, batsmen on field, overs, match status, target runs in second innings, playing teams, date, series title and match venue.

In order to make a system which is capable of working in a distributed environment, we incorporated a message queue into the system which can send data from one node to other in a cluster. Apache kafka[4] is one such open source message queue available for use, and is used in the system. The data extracted by the crawler is then formatted and input into the receiver end of the message queue.

B. Processing the data

Depending on the event being analyzed in this system, the amount of data that needs to be processed can greatly vary. Hence, we needed a data processing engine that can handle the processing of data on one or more cluster nodes if need arises. Apache Flink[5] is an excellent choice for this. Flink is an open source data processing engine capable of dealing with Big data. It allows us to process large amount of data at high-speed. It also provides support for both stream as well as batch processing.

The output end of the Kafka message queue is integrated with java program on Flink using Flink-Kafka connector that is available. The actual prediction of the end of innings run and other pattern discovery is done on Flink.

C. Score prediction for cricket match

We have used the ideas from method devised by Sankarnarayanan, Sattar, and Lakshmanan[1] for predicting the cricket match score. As our requirement is too predict the score for an ongoing innings in the match, this model suits us the best as it considers the instantaneous data for the match along with the historical records in order to make the score prediction.

Just like the proposed model, we have divided the ODI match innings of 50 overs into 10 segments of 5 overs each. These segments act as milestones for the current match progress. We try to find the closest situations from the historical dataset(past matches) for the current match in terms of the segment conditions and calculate the most probable runs that may be scored in the ongoing innings.

In order to find the most similar match situations from the past, we use the K- Nearest Neighbor algorithm for current segment data along with the historical data, which is also aggregated in terms of segments of 5 overs per match per innings. Here the value of K is taken as 5. The features used in order to perform K-NN are broadly divided as Instantaneous features that depend upon the current state of game and Historical data that are derived from the past match records. These features are inherited from the model explained in [1]. The distance measure used to compute similarity between feature vectors is Euclidian distance.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 02, February-2017

The algorithm for score prediction is described as follows:

Let X be the set of historical data available.
Let y be the feature vector (Instantaneous and Historical features) for current match.
Let n be the current inning segment.
Let N be the vector representing 5 nearest neigbours.
$N = \{n_1, n_2, n_3, n_4, n_5\}$
Let r_i represent corresponding runs at the end of segment n_i
Let R_n be the runs till current segment in the game.
Let R_{eoi} be the predicted runs at the end of innings.

Input: y,n
Output: R_{eoi}

$R_{eoi} = R_n$

```
for i = n + 1 to 10
        sum = 0
        N = NearestNeighbor(X,y)
        for j = 1 to 5
                sum = sum + r_i
        end
        runs = sum/5
        R_eoi = R_eoi + runs
end
```

D. Prediction of score in 2nd innings

For second innings, the nature of the match is different as the batting team usually either loses all the wickets before end of 50 overs or they successfully chase the target in less than 50 overs. As such many times, the team does not play a 50-over innings. To compensate for this fact, our system also tries to predict the wickets for batting team during the second innings, and hence only makes predictions for segments till which the team is still predicted to have wickets left.

E. Delivery of results

The results thus obtained are to be posted on the admin social media account. This is done using RestAPIs provided by the social media for developers. For Facebook we have used the GraphAPI for automatically posting the results on our Facebook page. Similarly for Twitter, we have used Tweepy, a RestAPI for posting the result on our Twitter page. The users that subscribe to our social media feeds automatically start receiving notifications as and when they are posted.

V. PROCESSING

For the prediction of score for a cricket match, we have a small sized dataset and limited algorithms i.e. K-NN and K-means clustering that are not very resource intensive. Hence, we perform all processing on a single machine with quad core processor and 4 GB RAM. All the processing is done on linux platform.

The prediction algorithm takes around 1-2 seconds per segment iteration. Overall, the system takes 2-4 seconds for extracting, processing and posting the results of match on social media using the web APIs.

VI. RESULTS

A. Dataset

Our dataset consisted of 145 ODI matches played in the duration of 1st JAN 2016 to 26 JAN 2017. Only the matches played between the 9 aforementioned ICC teams were taken into account. Also, the matches interrupted because of rains and other unpredictable factors were not considered. The ball by ball data for matches was scraped from www.espncricinfo.com. The team wise statistics and batsmen statistics are also extracted using the *statsguru* tool provided by www.espncricinfo.com.

Out of these 145 matches, 120 matches were used for training purpose and remaining 25 matches were used for testing.

B. Results for 1st innings

In the results we observe that our system works quite well after the 8 segments of a match. While it still gives acceptable results in around 50 percent of matches for R_{eoi} from segment 6-8, it still suffers in that duration because of the unpredictable nature of the sport. Good accuracy in score prediction is seen after around 40 overs down in the innings.

Figure 2 shows the results for prediction at the end of 8th segment while Figure 3 shows the results at the end of 9th segment.

As can be seen, for after 9 segments, more than 85 percent of times the end of innings runs are predicted with error margin of less than 20 runs.

Fig. 2. Prediction performance for 1st innings after 8th segment.

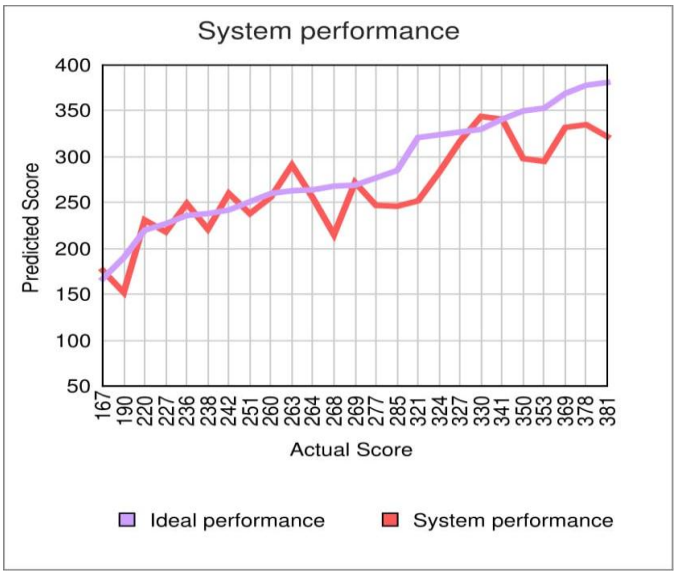
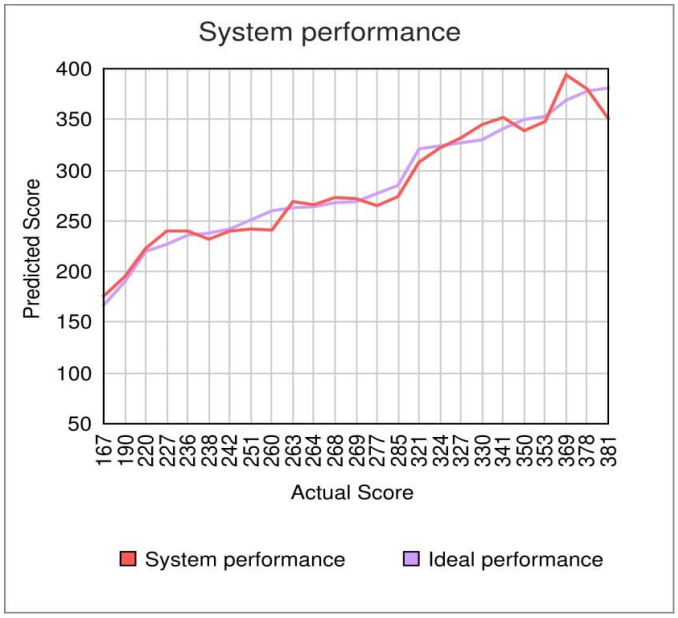


Fig. 3. Prediction performance for 1st innings after 9th segment.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 02, February-2017

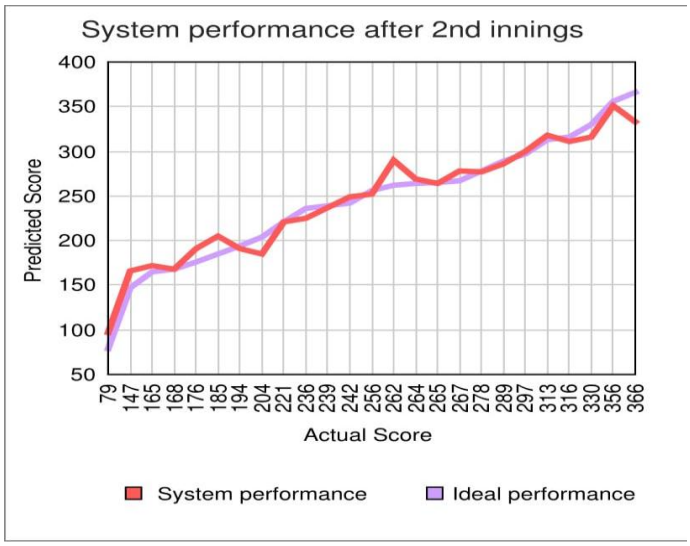


C. Results for 2nd innings

For second innings, we again observe a very good performance for last and last but one segments. Many times the team batting in the second innings does not get a chance to play all 50 overs, because of which the predictions does not consider 10 segments in all case. Instead it tries to predict the end of innings score R_{eoi} according to the prediction of how long the team keeps all its wickets or does not reach the target.

Figure 4 shows the system performance for prediction of runs in 2nd innings.

Fig. 4. Prediction performance after last second played segment in 2nd innings.



Hence, in 23 out of 25 test data matches, the winner/loser prediction has been accurate since after the completion of mid 2nd innings in real time.

D. Posting results on Twitter

The results are posted in form of text statements on our Twitter page. The overall latency of the system is 2-4 seconds after the occurrence of event.

Figure 5 shows some posts made by our system on Twitter during testing sessions.

Fig. 5. Final posts on Twitter.

VII. CONCLUSION

In this paper we have proposed a system that is capable of extracting live information about ongoing event, perform its analysis which can be of various kind depending upon the application and deliver the results to users via social media feed. In this case, we have presented a system that extracts live data about the ongoing cricket match from www.espncricinfo.com using a web crawler and tries to predict the total runs that may be scored in the innings by the team, hence commenting on potential winner/loser of the match.

Our system has proven to be quite robust with the match result prediction, giving correct results for 23 out 25 test set matches. The time taken for processing the prediction algorithm is approximately 1-2 seconds for each segment iteration. The results are immediately posted on the Twitter and facebook page using RestAPIs. The overall response time of the system is 2-4 seconds after the occurrence of any post worthy event.

VIII. FUTURE SCOPE

Future work for this system includes developing more bots for different events and to increase the number of analysis results provided for the existing events i.e. cricket match.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 02, February-2017

REFERENCES

[1] Vignesh Veppur Sankaranarayanan, Junaed Sattar and Laks V. S. Laksh-manan, Auto-play: A Data Mining Approach to ODI Cricket Simulation and Prediction, SIAM International Conference on Data Mining,2014
[2] Tejinder Singh, Vishal Singla, Parteek Bhatia. Score and Winning Predic-tion in Cricket through Data Mining. 2015 International Conference on Soft Computing Techniques and Implementations- (ICSCTI) Department of ECE, FET, MRIU, Faridabad, India, Oct 8-10, 2015
[3] Predicting a T20 cricket match result while the match is in progress. Fahad Munir, Md. Kamrul Hasan, Sakib Ahmed, Sultan Md. Quraish. Department of Computer Science and Engineering BRAC University, Bangladesh.
[4] https://kafka.apache.org/
[5] https://flink.apache.org/