Jenny Feng. Real-Time Event Summarization of Esports Events Through Twitter Streams. A Master's paper for the M.S. in I.S. degree. April, 2018. 50 pages. Advisor: Jaime Arguello.

Exploration of real-time summarization (RTS) methodologies and applications to esports events on Twitter. The goal of this study is to evaluate the effectiveness of real-time summarization techniques at esports event detection, highlight summarization, and timeline generation. A two-step system of event-prediction and summarization is proposed. First, using Twitter as the data source, events in an esports game are predicted through machine-learning and classification to determine event occurrences. Four major classification features and five standard classification models (Naive Bayes, Logistic Regression, Decision Trees, K-Nearest Neighbors, Support Vector Machines) are evaluated to identify an optimal event-detection model. Second, natural-language text processing functions such as term-frequency and TF.IDF are evaluated for effective event summarization and to confirm successful event-detection. The CART (classification and regression tree) model is selected as the most optimal model for event-detection, predicting in-game esports events with 70% accuracy. This study demonstrates the application of Twitter as a data source in detecting real time esports events.

Headings:

Social Media -- Communication

Classification -- automatic classification

Real-time computing -- Online chat

Content Analysis -- Communication

Response Rates

REAL-TIME EVENT SUMMARIZATION OF ESPORTS EVENT THROUGH TWITTER STREAMS

By
Jenny Feng

A Master's paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science
in Information Science.

Chapel Hill, North Carolina

April, 2018

Approved by

_____

Jaime Arguello

INTRODUCTION

League of Legends is a popular video game that holds live tournaments, which are broadcast to millions of viewers around the world. These broadcasts engage their viewers through video streams as well as social media platforms such as Twitter.

This study explores the application of real-time summarization to these scheduled tournament games using social media activity from Twitter. The goal is to explore automated event detection within a live data stream. Using machine learning to detect events that occurred inside this window, a summarized event timeline is generated.

Event detection and summarization are especially interesting for esports games which often encourages virtual audience participation through social media. Automating this process of event detection can provide more benefits than manual play-by-plays due to the speed at which the audience reacts to in-game events. An event can be described as something noteworthy or interesting happening in the game, much like a touchdown being scored in football.

For this paper data was gathered from the *Worlds 2017* League of Legends World Championships via a Python script and the Twitter API. Event-detecting models were then generated using Weka and Lightside and later summarized using natural language text processing.

The primary goal for the first half of this study is selecting the optimal model for event-detection. This process dives into the selection of various features for five standardized models to arrive at an optimal solution. Afterwards a secondary goal is to manually test the models against real world data, to validate their accuracy.

Background Concepts

Real Time Event Summarization (RTS)

RTS is a text-retrieval track that focuses on the process of recognizing and harvesting useful information from continuous digital data streams. Due to recent growing interest in these types of machine-learning systems, the RTS track at the 2017 Text Retrieval Conference (TREC) (TREC Real-Time Summarization Track Homepage) was created. The official TREC RTS track defines their goal as attempting to solve tasks in which the system automatically monitors data streams in order to alert users on flagged topics of interest - through, for example, push notifications.

A two-step system is proposed to detect and summarize events. The first step is the detection of an event in real-time. Detecting an event requires the system to classify moments in the data feed as significant or not. If an event is detected, the second step in the system will provide a descriptive summary. The end-result of this two-step process is an automatically generated timeline of events that maps to the actual events that took place.

Twitter

Twitter is a global and massively popular social media platform where users can interact via short messages called “tweets”. Twitter as a social media platform provides a real-time channel for focused discussion and allows users to stay updated on ongoing events. Its accessibility and

ubiquity make it a valuable source of big data as users frequently provide live information coverage on events.

Close to 100,000 tweets occurred within a 3-hour window during the 2017 world championship tournament for League of Legends. This overwhelming amount of data and information generated on Twitter cannot be processed by human standards. Given a Twitter stream alone, it can be difficult to parse out what important events happened during a game. However, the two-step RTS system will be able to use this content-rich data set to detect, highlight, and summarize events in real-time.

Esports

In recent years, esports has exploded as a cultural phenomenon, and the impact of its growth in digital entertainment as well as cultural significance has drawn the interest of many. The gaming industry has always held a major share of the casual entertainment sector, but esports is turning gaming from a hobby into a profession - and it has already begun to compete with some major traditional sports outlets for viewership.

It's no surprise that data analysts have jumped on the opportunity to capitalize on this new interest. In such a digitally founded world, all the data in the world is there for them to use. From miniscule details such as player profiles and match histories, to brand analysis and consulting for major esports teams, brands, platforms, and products - both the data and the demand for esports data analytics continues to grow.

But in terms of academic research, esports is a very new and still relatively unexplored field. Given its close similarities with traditional sports, one of the motivating factors of this research paper is to take applications of academic data analysis performed in traditional sports and apply it to esports to see if the results and conclusions still hold up in a similar but slightly different environment.

League of Legends World Championships 2017

The League of Legends World Championships 2017 (2017 World Championship) is an annually held professional League of Legends world championship tournament. It is the culminating event of two regular seasons held across multiple professional leagues that span the globe wherein the top teams from each of the global leagues are invited to compete for the championship title in the World Championship tournament. Because of the global popularity of League of Legends and it's dedicated fanbase it was certain to generate high traffic and discussion, moreover the easy accessibility of relevant data using established hashtags for Twitter filtering parameters and access to a gold standard for event comparison made it an optimal event to analyze.

RELATED WORK

This study is the first known attempt to combine RTS methods, esports, and social-media data streams. By demonstrating the successful implementation of the two-step system, the process can be applied not only to future esports streams, but other social-media driven events as well. Both RTS and esports are relatively young fields and there have yet to be any major academic inroads made in regards to research and study. Even fewer studies have attempted to combine

the two topics. However, there have been past studies that made significant contributions towards shaping the creation of the system.

Studies on RTS and RTS-related methods have been central in designing the two-step system. Furthermore, studies on understanding social media models, metrics, and user behavior patterns have aided in the design and implementation of the criteria during event detection. The biggest contributing studies to this paper were previous efforts at combining Twitter data and RTS machine-learning applications in the context of sporting events.

In their 2010 paper, Petrovic et al. (Petrović, 2010) first proposed the potential for Twitter as a platform for event detection and discovery. The first forays into event detection using Twitter data-streams were aimed at detecting major events. Sakaki et al. (Sakaki, 2010) was one of the earliest adopters of this concept and investigated earthquake detection through twitter monitoring. By using machine learning and creating a tweet classifier that was based on keywords, number of words, and context, they were able to create a model that was modestly successful at detecting earthquakes. Despite their success at creating an event-detecting model, Sakaki et al.'s model suffered from detection lag, which meant that they were not able to detect events in real time.

In their study on NFL games, the effectiveness of change, or increase, in tweet activity to detect an event was demonstrated by Zhao et al. (Zhao, 2011). By using Twitter as a sensor for detecting events in real-time, they were able to observe that prominent spikes in posting rate typically coincided with important game events. They then applied a lexicon-based classification system to identify different types of NFL-focused events. They were then able to successfully

build upon their system in order to detect and recognize NFL game events. With their goal of fast detection and recognition, Zhao et al. focused on real-time detection.

Hannon et al. (Hannon, 2011) looked at using summarization approaches with Twitter data in order to produce highlight clips of the World Cup. Punera and Chakrabati (Punera., 2011) were the first to utilize Hidden Markov Models in order to construct event-descriptions of events from Twitter data based on prior knowledge of similar events. However, both these studies failed to provide a critical aspect of RTS, which is event detection and summarization in real-time.

In their study of soccer game event detection in conjunction with Twitter data-streams, Zubiaga et al. (Zubiaga, 2012) sought to build even further on the success of Zhao et al.'s NFL event-detection system. This was done with a focus on creating an organically summarized stream as opposed to a pre-categorized event system. Zubiaga et al. proposed a two-step summarization approach composed of event detection, and then tweet selection for the Copa America 2011 soccer games. The success of their real-time event detection and summarization system was then referenced against sports journalists' live reports and found to be extremely efficient at detecting events in real-time.

Zubiaga's two-step process forms the basis of this study on esports Twitter data-streams and RTS methods. Similar to their detection and summarization for soccer games, this study utilizes a two-step method in order to propose a system for the detection of esports events in real time, as well as generation of a descriptive timeline of events.

METHODOLOGY

Data Collection

Due to its popularity and global audience, the 2017 League of Legends Worlds Championship is selected as the primary data source. A high volume of tweets is expected, making it an ideal target for data collection. These tweets are filtered by subject through hashtags, ensuring the collection consists of meaningful content. This data will be compared to the official end game summaries in order to validate the analysis.

The World Championship series consists of a best of five game series between the two finalist teams, SKT T1 Telecom and Samsung Galaxy, played on November 4th, 2017 (2017 World Championship). To mitigate the impact of extraneous variables as well as maintain consistency across activity, volume, and user-interest, data collection was limited to the finals series. The process of data collection itself was facilitated by the Twitter API (Twitter Developer) and the Tweepy (Tweepy) Twitter python streaming module. The Python script (see Appendix B) collected the tweets in real-time and then stored them in a SQL database. Data collection began one hour before the start of the series and ended an hour after the final match of the series.

Gathering the Raw Data

To ensure only relevant tweets for the event were collected, the Twitter datafeed was filtered using several hashtags unique to the World Championship series. The official

hashtags for this finals series were *#WORLDS2017*, *#SKTWIN*, *#SSGWIN*. Furthermore, any tweets directed at relevant Twitter accounts such as *@LOLESPORTS*, *@SKTELECOM_T1* were collected as well.

These event-specific filters guarantee that the majority of tweets sent out regarding the series would be flagged and collected. 86,668 tweets were collected over a 4-hour period from 3:03 AM EST to 7:14 AM EST. As seen in Figure 1, tweet activity between games varied, with the final game receiving the greatest number of tweets as it was match-point between the two finalists. A full timeline of the collected data is visualized in Figure 2.

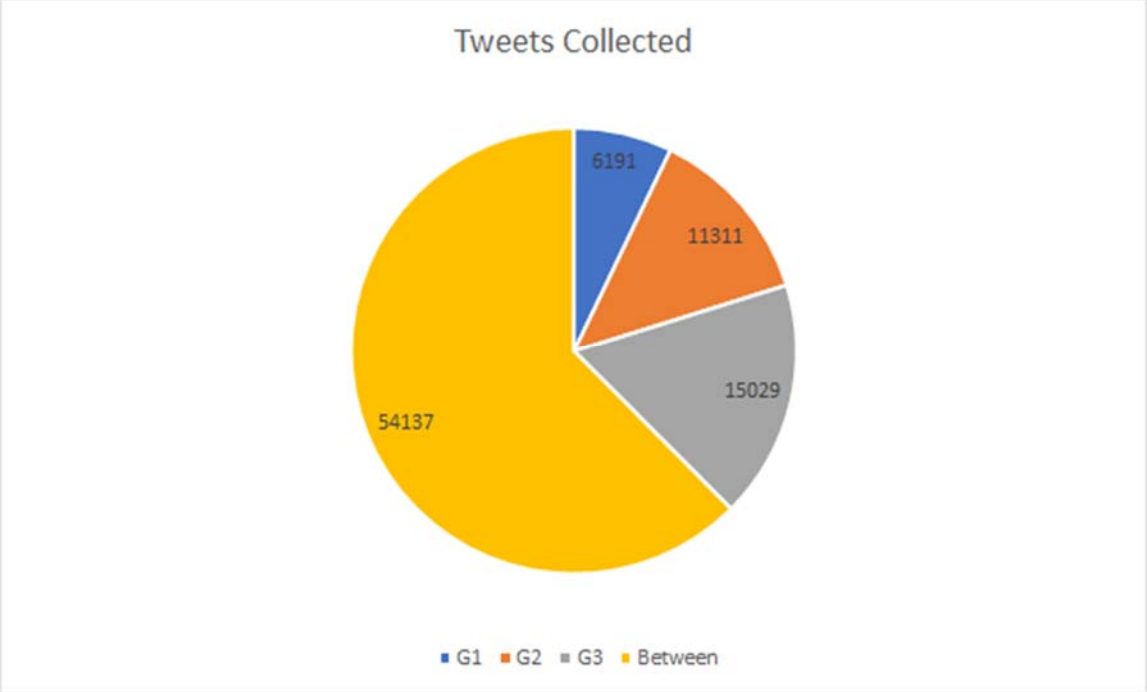

Figure 1 – A visual representation of all the Tweets gathered during the series and the fraction of Tweets actively made during each of the three games in the series.

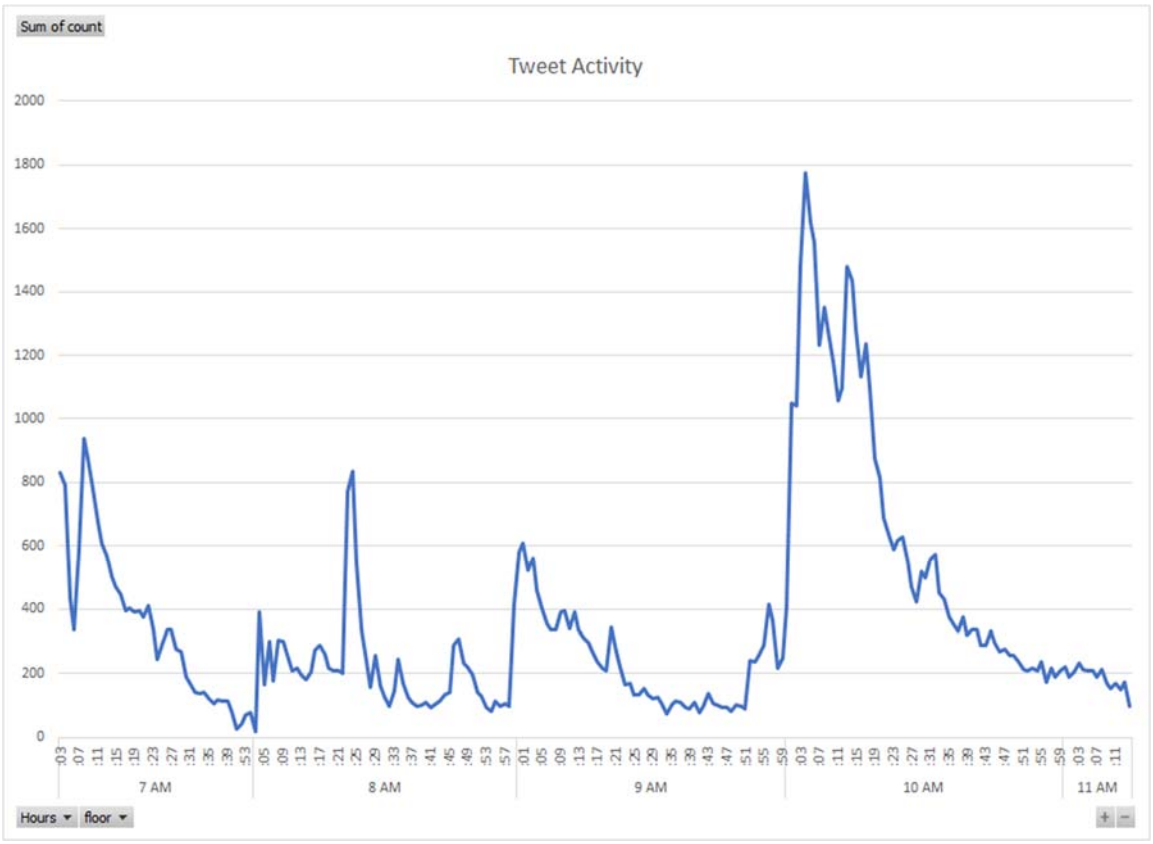

Figure 2 – A time series graph of overall Twitter volume made during the series.

Similar to goals in a soccer match, major events in a typical League of Legends game can be defined by various objectives. When one team achieves one of these objectives it pushes them closer to victory and is expected to trigger a spike Twitter activity. Major objectives for this study included: first blood, tower kills, dragon kills, Rift Heralds, Baron Nashors, and nexus destruction. For the purposes of the study, events are classified as game start, game end, and these major objectives. The end game summary previously mentioned is visualized by the official match history scoreboard. These events, as well as many more statistics are provided during the end game summary as seen in Figure 3.

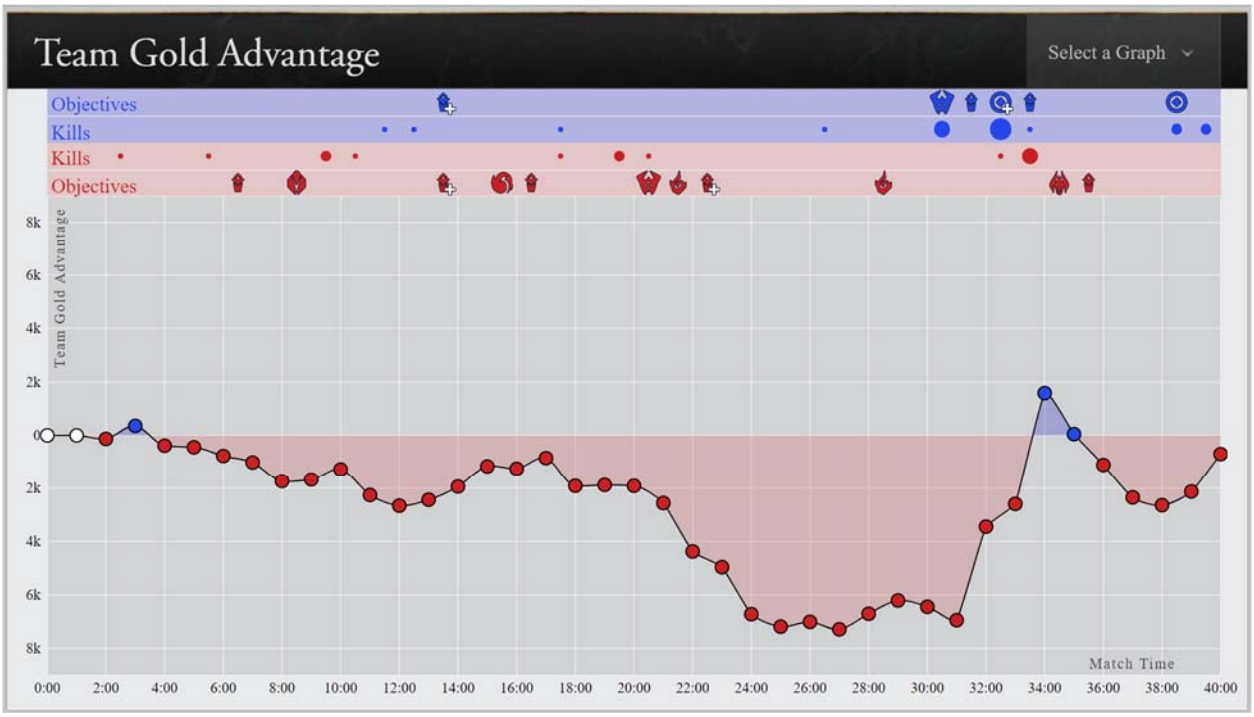

Figure 3 – The official end-game scoreboard. This is used to mark official event occurrences and classify instances for event detection.

Processing the Raw Data

The raw data gathered in the SQL database was exported to a CSV file. During this export the following attributes were considered: data timestamp, unique tweet ID, Tweet text, and hashtags. These attributes are necessary for either event detection or summarization. Other attributes that may be of interest in future studies include: retweet count, favorite count, and the other hashtags included in the body of the tweet as non-official hashtags can also often be used to identify the event at the time in which it was sent out.

Data Cleaning and Preparation

Further data cleaning and preprocessing took place in excel with the exported CSV file. The file was split based on the following event timeframes: pre-series, Game 1, Game 2, Game 3, post-

series and between-series. Each file contains collected tweets during those respective timeframes.

The Twitter activity baseline for the dataset of each game was established by extending the time-frame by 5 minutes before and after. The timestamp of each tweet sent was extracted from and formatted into the *hh:mm:ss* format. To calculate the outgoing Tweets per minute, the timestamp was rounded down into one-minute intervals. For each Game, a pivot table was generated that organized the data by one-minute intervals and the number of tweets in that interval (Figure 4). One-minute intervals was deemed the best as it allowed for lag-compensation between an event and the time it takes to compose and send a tweet (Zhao, 2011).

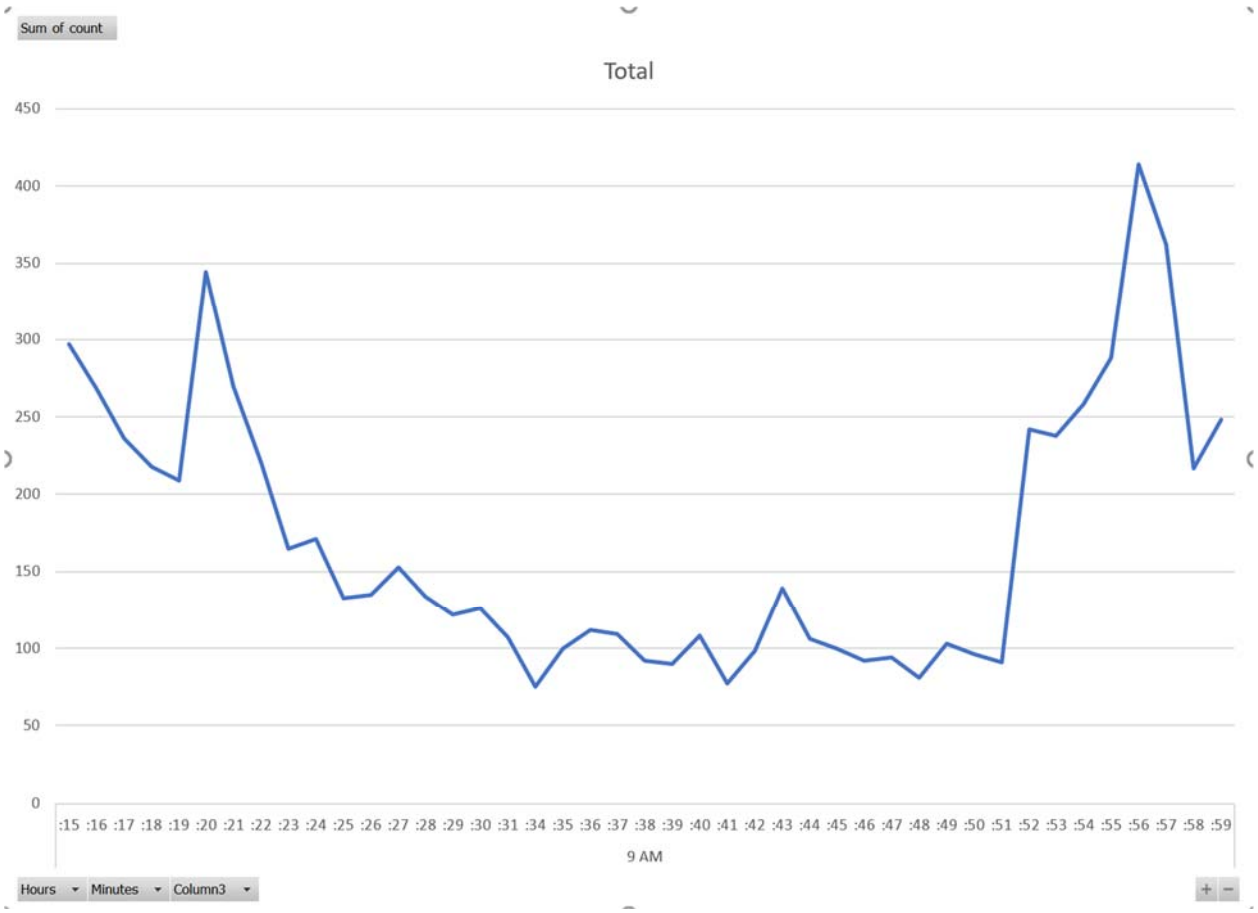

Figure 4 – A time series graph representing Game 3 Twitter activity during the course of the game.

The result of this process was a series of condensed datasets representing Twitter activity per game minute. Each minute interval constitutes an instance, and the resulting number of instances available per game yielded: Game 1 with 48 instances, Game 2 with 46 instances, and Game 3 at 52 instances.

Event Detection

The first step of the two-step process is event detection. To create a model for event-detection, several tasks need to be accomplished: evaluation and selection of appropriate features for event classification and training, evaluation and selection of a model that can provide the most

accurate results, and finally successful real-time event detection using the machine-learning model.

The java-based machine learning software WEKA (Weka 3) was used in conjunction with the open-source Lightside (Lightside) machine-learning platform to perform the majority of data training and analysis.

Class

The initial event detection runs through all known instances (the 1-minute timesteps for each of the three games in the series). In every instance there are two options; an event occurred or an event did not occur. For training and accuracy analysis, every instance was manually referenced against the official summary and retrospectively labeled using gold-standard binary event labels. Using these binary classifier, instances were labeled as TRUE (an event occurred) or FALSE (an event did not occur). These TRUE and FALSE binary labels were chosen to provide easy visualization and comprehension.

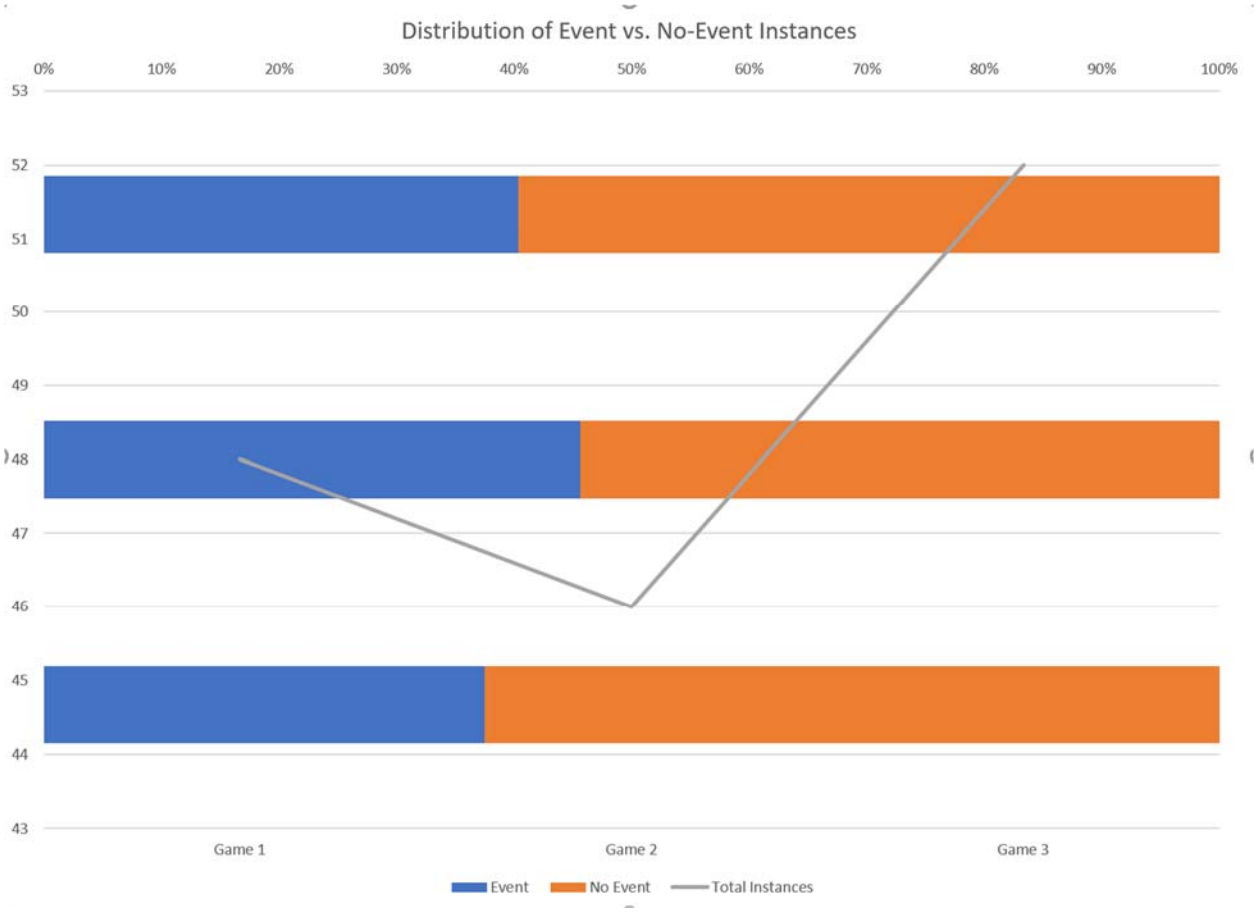

Figure 5 – Graphical representation of the distribution of instances labeled as TRUE or FALSE per game based on the official end-game scorecard.

Models

As mentioned previously, event-detection requires models to successfully classify and detect events. Each of the five standard classification models has its own methods. These five different types of models were compared for event-detection: Naive Bayes (NB), Logistic Regression (LR), Classification and Regression trees (CART), K-Nearest Neighbors (KNN), Support Vector Machines (SVM).

Models were tested using the basic settings in WEKA, plus the following parameters:

- Naive Bayes model - no kernel estimator or supervised discretization was used in the building of the Naive Bayes model.
- Logistic Regression model - was set to run until it was estimated that the algorithm converged with default ridge estimator regularization.
- Classification and Regression Tree model - depth of the tree was set to be automatically defined, pruning of the tree was allowed, and the minimum number of instances supported by the tree in a leaf node when constructing the tree was set to 1 instance.
- The K-Nearest Neighbors model - the best value of K was found by allowing the model to cross-validate to find the best value of K, and distance was measured using a linear Euclidean distance evaluation method.
- Support Vector Machine model - the complexity parameter was set to 1 to allow for margin violations, and a polynomial kernel was used to separate the classes to allow for greater flexibility in the model as well.

By evaluating the models using the classification metrics (kappa, accuracy, precision, recall, and F-measure), we compare these models in order to determine which type of classification model would be the most effective in detecting the occurrence of an event.

For each model, several evaluation metrics were recorded:

- The Kappa statistic, which represents the degree of agreement (Banerjee, Capozzoli, McSweeney, & Sinha, 1999)
- The accuracy, which is the number of correctly identified instances out of the total number of instances (Taylor, 1999).

- Precision with respect to an event occurring (TRUE), is the fraction of relevant instances among the retrieved instances. A max precision score of 1.0 indicates that every instance classified as a period during which an event occurred, was correctly classified. However, regarding the number of incorrectly labeled instances, it says nothing (Powers, 2011).
- Recall with respect to an event occurring (TRUE) is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. A max recall score of 1.0 indicates that all instances classified as an event were correctly classified. But it says nothing of the number of incorrectly classified non-event instances (Powers, 2011).
- The F-measure with respect to an event occurring (TRUE) takes both precision and recall into account in order to determine the harmonic average of both values. A perfect F-score of 1 indicates perfect precision and recall (Powers, 2011).

Features

All features were derived from twitter traffic. Previous studies have demonstrated the effectiveness of using increases in Twitter activity to detect the events, so the features selected for this study are also built upon the concepts of Tweet volume. Next, these selected features were evaluated and selected in order to build the most effective models.

Each instance in the datasets was accompanied by a classifier and four main features:

- Sum: the sum of the number of outgoing tweets during that instance (1-minute time-step).
- Game time: the match time of the game in minutes (with 0 being the start of the game, -1 for a minute before the game start, and 1 for the first minute of the game).

- Delta: the change between the current instance and the previous one.
- Percent change: and the percentage change in the number of tweets per interval (the delta divided by the sum of the first instance).

The sum and game time are basic features that map directly to the available data whereas the delta and percent change are derived from these two basic features. This approach supports the concept that an important event will trigger a sudden increase in relevant tweet activity, and that will be reflected by the increase or decrease in the sum of the tweets (Sakaki, 2010).

The percent change is the relative change between the sum of the outgoing tweets and represents the magnitude or extent to which the number of tweets being made changed over each instance. As percent values, the possible values ranged from [-100.00%,100%] with negative and positive values to indicate directionality and the percentage to indicate the magnitude of change. Percent change builds upon the concept of the change in sum feature, but instead only takes relative change from one instance to the next into account, and thus can mitigate potential wrong learning based on pure, raw numbers such as change in sum.

Feature Comparison

A feature ablation study was performed to determine which features would be purposive in creating an effective machine-learning model. Each of the 4 available features: Time in game, Sum, Delta, and Percent change was systematically removed and the Game 3 training set was evaluated against the 5 standard classification models using 10-fold cross-validation.

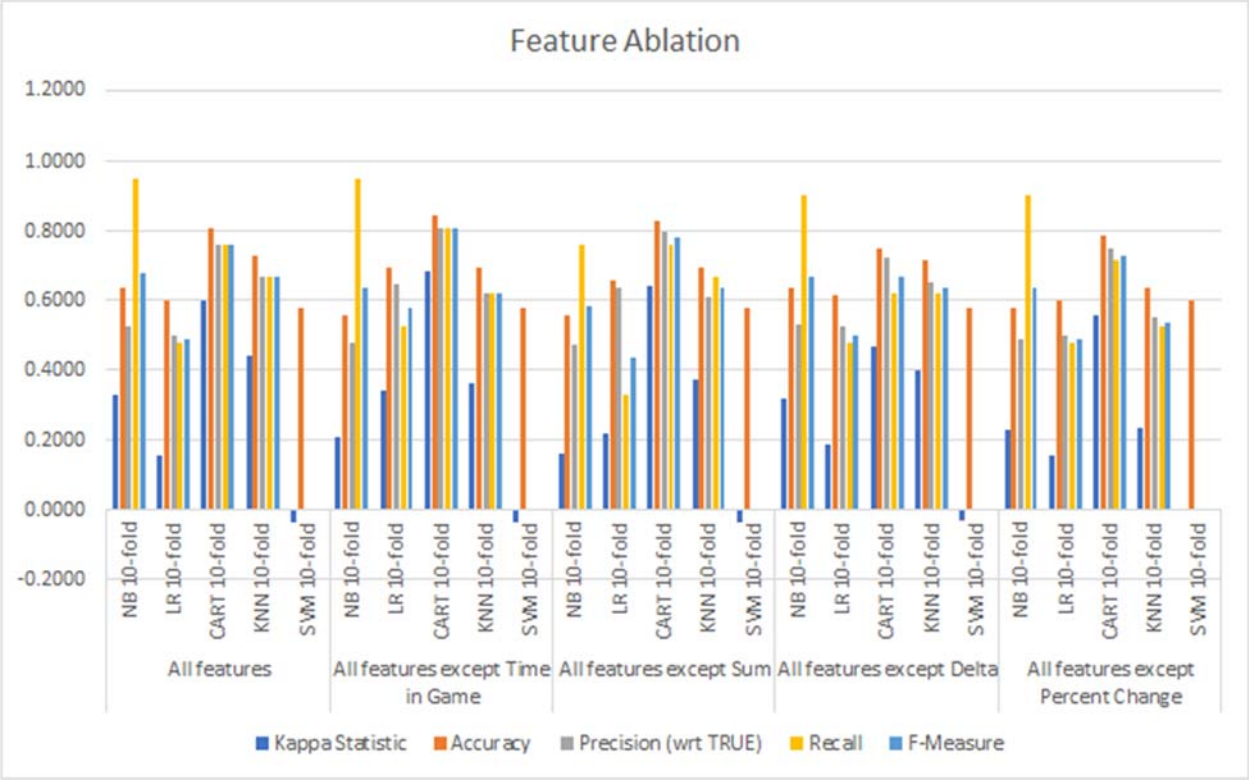

Chart 1 – Graphical chart depicting the results of the feature ablation test. Each of the 4 selected features was systematically removed and evaluated against the five standard classification models. Removal of features that resulted in poorer overall performance indicated the importance of that feature.

Feature Analysis

The results of the feature ablation test varied across model types, but on average the basic features, time in game and sum appear to contribute less to the success of the model than the derived features of delta and percent change.

The time in game feature is one of the two basic features that denotes the amount of time a game has progressed. From the results, this appears to contribute the least to successful event detection as all the models performed better on average even without it. While this feature allows models to successfully detect game start and approximate game end events, in regards to other in-game events the time in game feature is otherwise independent. Therefore, there may be a case for the removal of the time in game feature.

The sum feature is the second of the two basic features and is the sum of all outgoing tweets made during an instance. Removal of the sum feature produced more varied, but approximately similar results when compared to results using the sum feature. Accuracy was marginally improved, but recall metrics were poorer in comparison, so no conclusions can be drawn as to whether the sum feature has measurable value in successful event detection. Since the sum is the raw value of tweets per instance, the feature is inherently unique to the dataset and may inaccurately influence training through this bias. However, as event-detection is ideally self-contained without requiring prior knowledge, the actual impact of the sum feature on performance is uncertain.

The delta feature is a derived feature that calculates the change in sum between once instance and the next. Without the delta feature, model performance decreases across all models, indicating that it is a major contributing feature to successful classification and event detection. Likewise, the derived percent change feature which denotes the percentage change in sum, also shows poorer event-detection results when compared against models using all features. Comparing the two derived features, it appears that the delta feature is more beneficial towards the creation of a classification system for detecting events when evaluated alone. However, when compared against the results given when using both the delta and the percent change in conjunction with the two other basic features, the resulting model yields better results.

From the feature ablation test, it was determined that the two basic features of time in game and sum may potentially be inhibiting event-detection, but both derived features of delta and percent change are necessary features for model success. Together, they synergize to benefit the system as a whole, and create a better event detecting model.

As will be confirmed later during model evaluation, the CART model yielded the best average performance regardless of feature selection.

Model Analysis

To evaluate the five standard classification models on successful event-detection,the dataset for Game 3 was chosen as the training dataset. Since the Game 3 dataset contained the greatest volume of tweets and tweet activity, as well the greatest number of instances due to it being the longest game in the series it was determined to be the most appropriate.

Furthermore, due to the inconclusive results of the feature ablation test, two separate model evaluations were run. The first test is an analysis of the models using all four features (Time in Game, Sum, Delta, Percent Change), and the second test evaluates only the two derived features (Delta, Percent Change). These will be referred to as the 4-feature test and 2-feature test below.

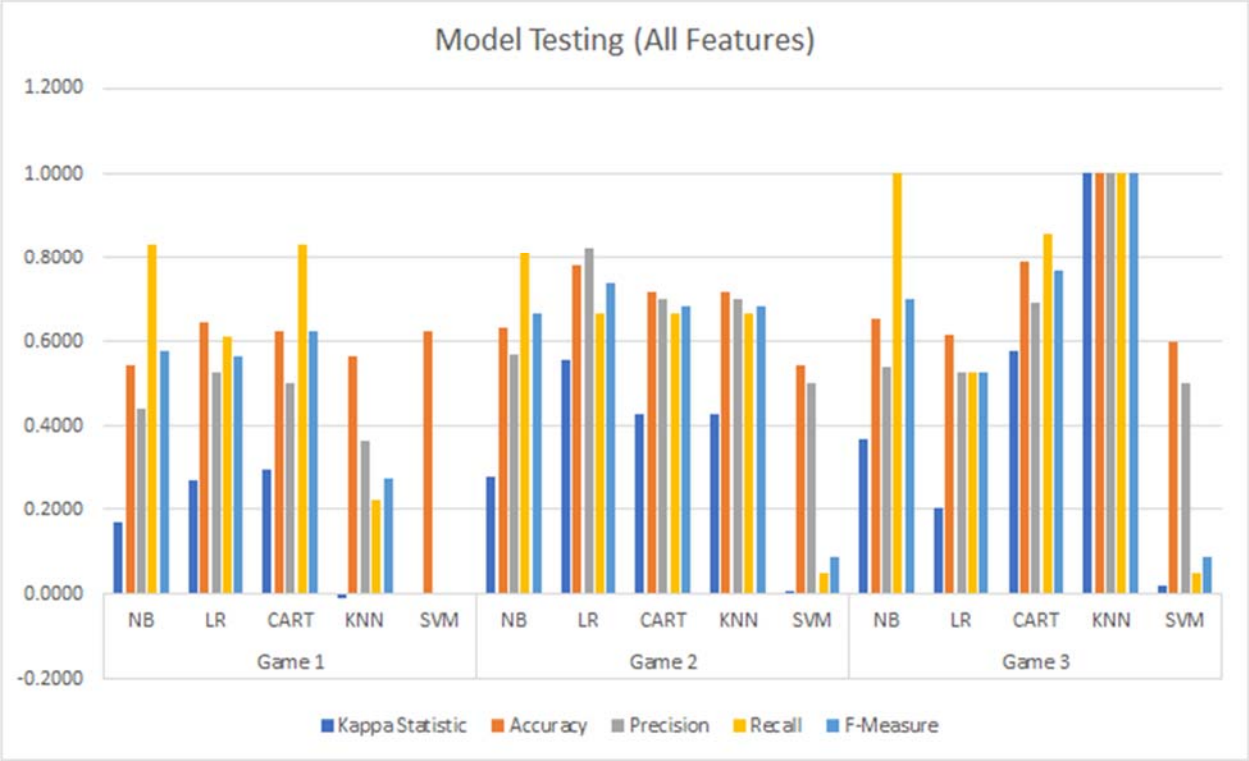

Chart 2 – Chart depicting the evaluation results of the five standard classification models against novel datasets (Game 1, Game 2) using all 4 features (Time, Sum, Delta Percent Change).

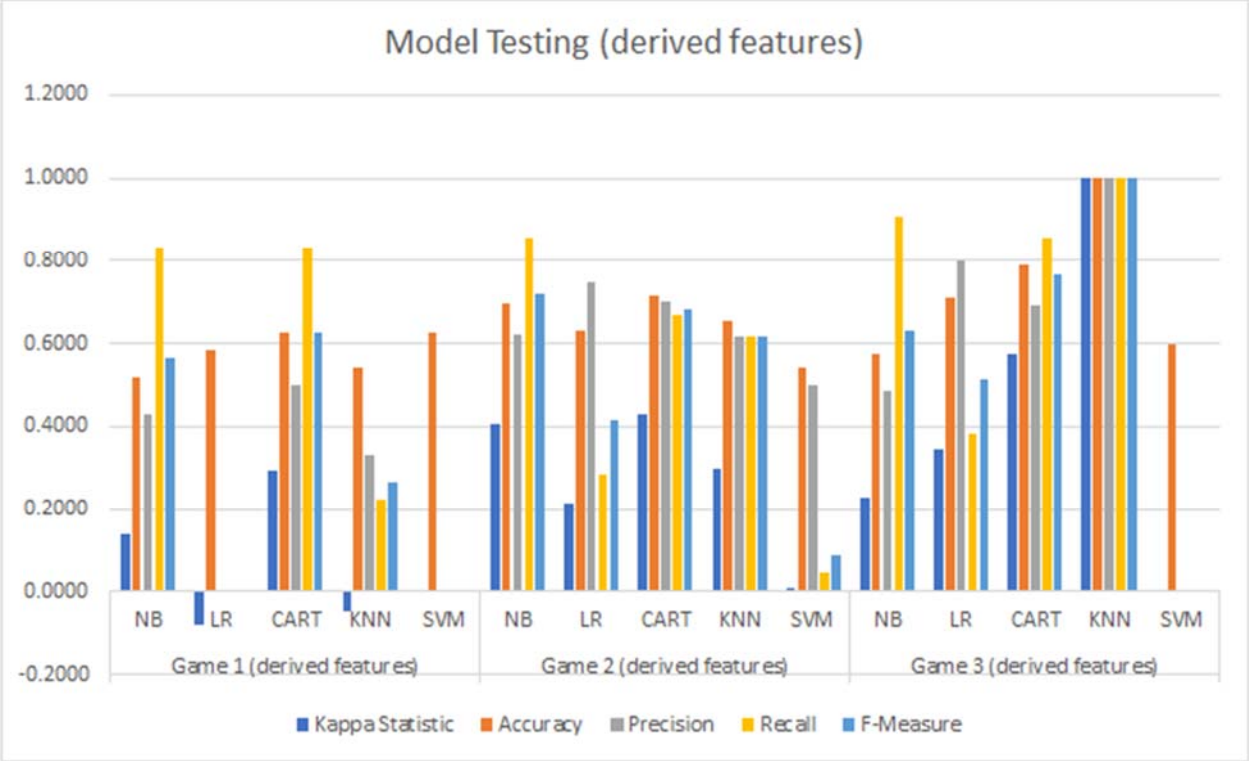

Chart 3 - Chart depicting the evaluation results of the five standard classification models against novel datasets (Game 1, Game 2) using only derived features (Delta, Percent Change).

From the initial comparison of the results across all the datasets, it was determined that the CART model was the most appropriate model out of the five standard classification models.

The Bayes Theorem describes the probability of an event based on prior knowledge of conditions related to the event - Naive Bayes is a simplified implementation of this theorem that assumes conditional independence between features. Across the three Game datasets, the Naive Bayes model performed consistently but underperformed compared to other models. Despite average performance in terms of Kappa statistics, Accuracy measures and F-measure, the Naive Bayes model consistently outperformed other models in terms of Recall metrics regardless of which features were used. This indicates the Naive Bayes model was able to successfully identify almost all events, but unfortunately also misidentified many non-event instances as events as well. Although it was consistent in its results, the model performance is considered average due to the low Precision and F-measure scores.

Logistic regression is a rather simple technique that utilizes regression techniques in which a coefficient is learned for every instance, combined into a regression function, and then transformed into a logistic function. The Logistic Regression model assumes a Gaussian distribution for the input variables, but it was tested regardless of the lack of a bell-curve in the representation of the training data. The LR model performed unusually well in the first 4-feature test for Games 1 and 2, but performed extremely poorly in the 2-feature comparison. The varied and therefore inconclusive results for this model are most likely due to the fact that there were only relatively few instances to work with, and dataset does not follow a bell-curve distribution.

CART (Classification and Regression Tree) models are also known as decision tree models. This model creates decision trees with branching nodes to evaluate instances by choosing the best split at each point until it reaches a prediction. Given the non-clustered nature of the classes, the CART model was expected to perform the best in both the 2-feature and 4-feature tests. In the 4-feature test, although the CART model had lower performance results than LR for the Game 1 and Game 2 datasets, it performed better on average across all three games. Moreover, given the LR model's poor performance in the 2-feature test, the CART model clearly outperformed all the other classification models there.

As the CART model performed the best overall in terms of accuracy, precision, recall, and F-measure metrics relative to other models, it was flagged early on as a candidate model.

The K-Nearest Neighbors model algorithm utilizes both classification and regression by searching for the k-closest known instances using a specified distance computation in order to predict the class of a new instance in the instance space. In the KNN model that was used, K=1 was found to be the optimal value of K, and so classification of new instances was predicted by the nearest single neighboring instance. K=1 was an appropriate value of K for the modeling of the datasets due to the low number of instances in the dataset. The KNN results for the Game 3 dataset are disregarded for both the 4-feature and 2-feature tests. KNN model results were too inconsistent between the Game 1 and Game 2 datasets regardless of features, and so it was ruled out as a potential model due to the high levels of variance. Further parameter tweaking and a larger dataset may help to improve the accuracy of the model as larger values of K can be tested, and different weighting and averaging systems can be used. But too much model optimization for the training set can also result in model exclusivity. As a result, the instability of the KNN model and the need for model flexibility for different datasets prevented it from being selected as the

results varied greatly relative to different cross-fold parameters for testing (as seen from the results of self-testing).

Support vector machines are based on the concept of optimization and finding a line of best-fit that can separate the data into the two binary classes. SVM finds a decision surface that maximizes the margin between the separation of the two classes by making use of the support-vector instances closest to the decision hyperplane to find the optimal solution. Regardless of which features were used or which dataset was tested, the SVM model performed extremely poorly. This is most likely a result of the difficulty of cleanly grouping the two classes which is only exaggerated by the small data set.

CART had better average performance when evaluated with all four features. Disregarding the KNN outlier, the LR and CART models were the top two performers when evaluated with all four features. While the LR model had better results for the Game 2 dataset, the CART model had much better average performance across all three games. The potential cause of this is likely due to the fact that CART is a non-linear classifier, which gives CART more freedom to exploit complex feature interactions within the decision tree. The NB, LR, and SVM models are all linear classifiers which makes it difficult to cleanly divide and identify classes within the feature space. While the KNN model is also a non-linear classifier, it was most likely limited by the relatively small dataset. Ultimately, the results of the model evaluation support the initial assumptions that the CART model would yield the best results.

Model and Feature Selection

The CART model performed the best on average in both the 2-feature and 4-feature tests. As a result, the CART model was chosen as the optimal model for testing against new data. Furthermore, given the results of the data, it appears that the impact of features on the CART model is low due to the fact that it was able to yield similar performance metrics in both the 2-feature and 4-feature tests.

Averaging the results of event detection across all three games, we see the model has approximately a 70% event prediction accuracy.

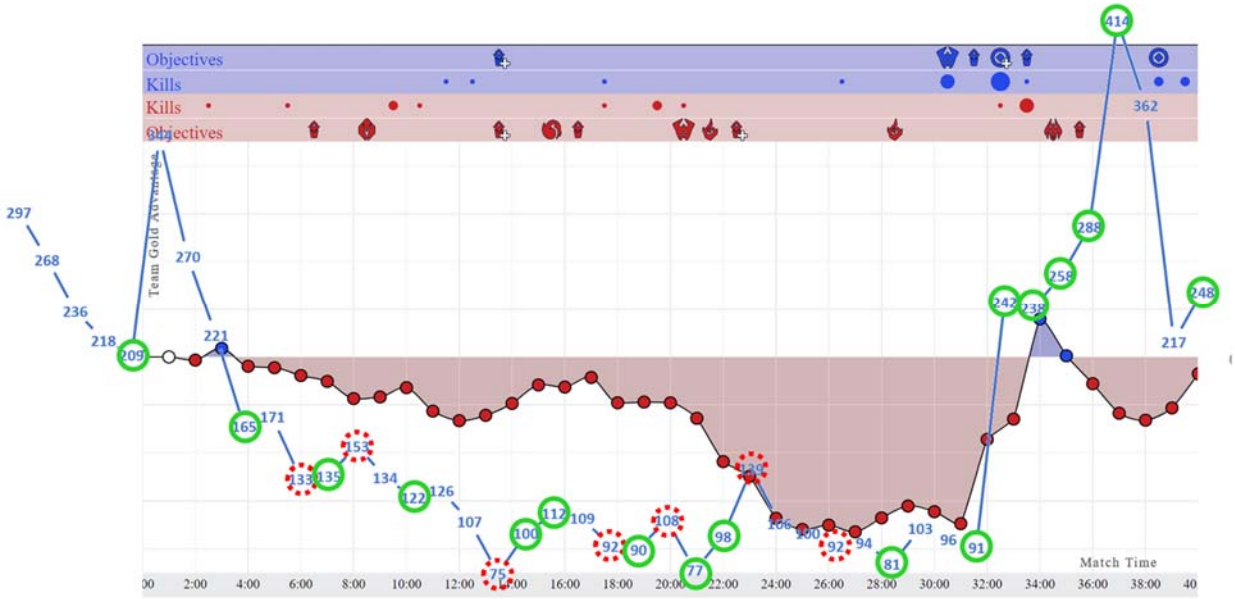

Figure 6 – Game 3 Tweet volume overlaid against the official Game 3 scoreboard. Each Game 3 instance is marked by the sum of outgoing tweets made during that instance. False positives are marked with dotted red circles, and correctly identified instances are marked with full green circles.

In the above graph (Figure 5), the official Game 3 end of game report is overlaid with the Game 3 dataset. Red dashed circles indicate either false-positive or false-negatives, and solid circles indicate correctly predicted events using the CART model and all four features.

The most common confounding factor in accurate event prediction appears to be lag. The lag between the time an event occurs and the time Twitter users actually tweet about it was only marginally taken into consideration in the use of 1-minute intervals for each instance. In reality, tweet composition and retweeting can often lag behind the actual event occurrence by several minutes. This is most readily seen in the fact that increases in tweet activity sometimes occur past the 1-minute interval in which event has officially been recorded. As a result, many of the false-positives of the model occur right next to instances in which an event has occurred but no major twitter activity has yet taken place due to lag.

While not perfect, more data and further training may be able to improve the accuracy of the event detection even further.

Highlight Summary

Once the first step detects an event, the second step analyzes the contents of the outgoing tweets during that instance. This analysis results in a description or summary of the detected event. Merely detecting the event is insufficient, the system should also be able to provide a descriptive summary in order to provide detail and context in its application.

Unlike event-detection, the highlight summary step relies only on one primary data source, the content of the relevant tweets for that detected event. By using natural-language text processing, term-frequency, and TF.IDF functions a summary of the event can be generated.

The event summaries created in step two are applied in two different contexts. Similar to the official end game summary a timeline can be generated using the event summaries. However, unlike the official end-game summary, which only provides game-relevant statistics, this offers insight into social engagement during a game. Another application of the event summaries is providing live-alerts and notifications to subscribed or interested users.

Term Frequency

Term-frequency is the number of times a word occurs in a body of text. To determine the details of an event, mining the most popular words tweeted using term-frequency is a straightforward summarization approach. The process for determining term-frequency is relatively simple. For an instance where event-detection occurs, the text contained within all the relevant tweets made during that instance can be aggregated into one complete document. Every unique word tweeted during the instances is counted to create a word corpus. Term-frequencies for each available word are ranked to yield the list of the top, or most frequently tweeted words, during the event. Stopwords can also be added to mitigate the impact of commonly used words as well.

Selecting the data

For this analysis, only tweets from a detected event from the Game 3 dataset were considered. The text from the outgoing tweets during that instance were extracted and saved as a single document in a CSV file.

Determining Term Frequency

From the tweet text, a tweet corpus was created. A corpus is the collection of unique words in a document. To calculate term frequency, a python script was written. This script read the text from each tweet made during this instance, added each new word it encountered to a dictionary of words to form the tweet corpus, and then counted every occuring instance of the word as an accumulating sum. This was ordered into a ranked list of words along with their corresponding term-frequencies.

Evaluating Term-Frequency as a Highlight descriptor

During the selected event, the team Samsung Galaxy defeated SKT T1 Telecom in a final teamfight to win both the match and the series in the League of Legends Worlds 2017 Championships.

The term-frequency tables below were evaluated to see how closely they reflect the actual occurrences of the event.

Word (with stopwords)	**TF**
#Worlds2017	264
#SSGWIN	78
RT	75
SSG	46
GG	34
a	28
SKT	27
the	24
in	23
@lolesports:	22

Table 1 – Table representing the top 10 words ranked by Term-Frequency for the given instance (stop words included).

From the above table (Table 1) of term frequencies, terms related to Samsung Galaxy take two of the top five TF spots. Moreover, we can infer that the game ended, due to the fact that the term “GG”, which is a common gamer and esports related abbreviation for “good-game”, appears as the 5th ranked term. Combining these two facts, it can be assumed that Samsung Galaxy won the game during this event.

To provide a more focused comparison between the event summary and the actual event, further processing steps were taken to clean-up stop words.

Commonly used stop words as well as Twitter short-hands and related Twitter accounts do not add value to the summary and were removed during this clean-up. An example of this is the #Worlds2017 hashtag, which served as a major hashtag for the overall series, as well as core search parameter in the event detection step, but does not provide any contextual information for step 2.

Word	TF
#SSGWIN	78
SSG	46
GG	34
SKT	27
fight	20
he	19
died	19
Faker's	18
face	18
after	18

Table 2 - Table representing the top 10 words ranked by Term-Frequency for the given instance (stop words removed).

With the stop words removed, the summary is more reflective of the actual event. #SSGWIN and SSG still hold the top-ranked spots to tell the story of Samsung Galaxy's win - but the words below could now be used to give some further insight into what else happened. The words "fight", "died", "Faker's", "face", "after", may seem nonsensical and ambiguous to a casual observer, but in context of the actual event, it can easily be pieced together to form a coherent sentence "Faker's face after he died in that fight". In this context, Faker is the moniker for one of the premier professional players on the team SKT T1 - and upon losing a major team fight, Faker's avatar was killed and Samsung Galaxy were able to take the win. While not a fully fleshed out description, those few words are enough to give any fan following the event a relatively good idea of the occurrences of the event.

Unsurprisingly, that exact sentence also composes the major parts of the most retweeted tweet made during that instance "RT @aliexk: Faker's face after he died in that fight". The implications of this will be discussed in the tweet selection method process below.

Word Cloud

A more visual representation of this list of term-frequencies can be created using a word cloud generator. This medium utilizes the same resources as the term-frequency table but represents the ranked TF list in a more visually pleasing and digestible format.

Figure 7 – Word cloud representation of the top 10 words ranked by Term-Frequency for the given instance (stop words removed).

TF.IDF

Event highlights can also be created using TF.IDF (term frequency-inverse document frequency). As TF.IDF measures the relative importance of each word in a tweet to the overall collection of tweets from that instance, it can provide a more accurate list of words important words as opposed to raw term-frequency.

To determine which words in the overall collection scored the highest TF.IDF, a python script was written to calculate the TF.IDF for all words contained in each tweet. The words with the highest TF.IDF values for each tweet are then collected and output into a pivot table that sums the number of times a specific word received the highest TF.IDF score. This produces a ranking table of the number of times each word received the highest TF.IDF score in their respective tweet.

Word	# Times Ranked Top TF.IDF Score
WORLDS2017	72
SSGWin	50
Gg	36
ssg	22
after	18
aliexk	18
face	18
CLOSE	16
SO	16
t.co/95AqPorv7s	16
skt	15
3-0	11

Table 3 - Table representing the top 10 words ranked by TF.IDF for the given instance (stop words included).

The TF.IDF ranked list end results are quite similar to the generated term frequency list. However, there are some discrepancies such as “SO” and “CLOSE”, and “3-0” which were not seen in the top 10 for term frequency. This was a result of the tweet “SO CLOSE” being retweeted a great number of times over the course of the instance, and since these are the only two words contained in the tweet, they appeared as the highest TF.IDF scores for each time the tweet appeared.

Individually, both ranked TF and TF.IDF lists are only capable of representing part of the full details of the event - but by combining the results of both the term-frequency and TF.IDF word ranking lists, a better summary can be produced.

In the table below, the resulting TF and TF.IDF event summaries are listed for three different instances.

Instance	True Class	Predicted Class	TF - Top 10	TF.IDF - Top 10
45	TRUE	TRUE	#WORLDS2017, #SSGWIN, RT, SSG, GG, a, SKT, the, in, @lolesports	#WORLDS2017, #SSGWIN, GG, SSG, after, aliexk, face, CLOSE, SO, SKT
		(removed stop words)	#SSGWIN, SSG, GG, SKT, fight, he, died, Faker's, face, after	
6	TRUE	TRUE	#SKTWIN, #SSGWIN, #WORLDS2017, @lolesports, One, greatness, New, Project, Skins, against	#WORLDS2017, game, a, SKT, to, from, win, of, and, away
25	FALSE	TRUE	Legends, Never, @lolesports, IT, SKT, #SKTWIN, #WORLDS2017, ACE, Baron, Dragon	#WORLDS2017, @SKTELECOM_T1, @LOLESPORTS, #SKTWIN, the, Never, a, Legends, game, Die

Table 4 – Table representing the top 10 ranked TF and TF.IDF words for 3 different instances. Instance 45 and instance 6 are both correctly identified events, while instance 25 is an example of a summary for a false-positive.

Instance 45 is a correctly identified and summarized event, as discussed previously.

Instance 6 shows a correctly detected event; however, the resulting frequency summaries are nonsensical. More context is required to accurately summarize the event. In this case, instance 6 represents the start of Game 3 and as discussed earlier, it suffers from lag in which the audience response is not contained within the appropriate instance.

Instance 25 is an example of a false-positive in which no event occurred but was identified as such. As expected, the summary is equally nonsensical. Audience participation during this instance is a result of delayed retweets and trivial content.

Instance 6 demonstrates potential room for summarization improvement. This can be accomplished through Tweet scoring and Tweet selection. The TF and TF.IDF rankings can be applied to Tweets to score them and create a list of ranked Tweets. The top-scoring Tweets can then be more heavily weighed during event summary. These top-ranked Tweets can provide better context as they are fully realized thoughts or phrases. Context can sometimes provide detail that words alone cannot, and a representative tweet instead of an abstract and deconstructed description view of the event with discrete words may be an alternative worth exploring.

Conclusion & Discussion

The goal of this study was to explore the application of real-time event detection and summarization as a means of detecting esports events through Twitter data streams. We proposed a two-step system composed of 1: classification-based event detection without the requirement of prior knowledge, 2: event summarization using text analysis. Together, this system can process events in real-time and to generate a social timeline or provide live notifications.

We were able to obtain approximately 70% event-detection accuracy when compared to official event reports, and initial event summarization has shown positive results. Thus, we have successfully demonstrated the potential for Twitter as a data source for real-time event detection in the context of esports streams and events.

However, after manual comparison of the model results against the official timeline, the presence of false-positives and false-negatives indicate that further improvements can be made. This can be addressed with a larger and broader dataset may have been introduced due to the dataset being limited in scope to a single tournament series. Furthermore, a better machine-learning model can be created using a larger dataset. Lastly, more advanced feature and parameter analysis can be applied to optimize the model.

Future Work and Application

Given the two-step process of successful event detection and summarization, there are several noteworthy real-world applications. In traditional sporting events, exciting moments are manually delivered to subscribers through live alerts. The system put forth in this paper is automated and therefore able to do so without this manual step. A secondary application is the visualization of the user engagement in a timeline during a game. Allowing for post-game analysis as well as interesting real time predictions of interesting events.

In future work, further improvements to the system such as: enhanced event-detection accuracy by means of more data and more training, more advanced summarization methods, as well as further integration of the overall process into a user-friendly UI would be ideal.

Appendix A

Feature, Model Tests and Results

(see next page)

A1. Feature Ablation Results

Features	Model Type	Kappa Statistic	Accuracy	Precision	Recall	F-Measure
All features	NB 10-fold	0.3288	63.46%	0.526	0.952	0.678
	LR 10-fold	0.1548	59.62%	0.500	0.476	0.488
	CART 10-fold	0.6006	80.77%	0.762	0.762	0.762
	KNN 10-fold	0.4409	73.08%	0.667	0.667	0.667
	SVM 10-fold	-0.0381	57.69%	0.000	0.000	0.000
All features except Time in Game	NB 10-fold	0.2090	55.77%	0.476	0.952	0.635
	LR 10-fold	0.3407	69.23%	0.647	0.524	0.579
	CART 10-fold	0.6805	84.62%	0.810	0.810	0.810
	KNN 10-fold	0.3610	69.23%	0.619	0.619	0.619
	SVM 10-fold	-0.0381	57.69%	0.000	0.000	0.000
All features except Sum	NB 10-fold	0.1648	55.77%	0.471	0.762	0.582
	LR 10-fold	0.2213	65.38%	0.636	0.333	0.437
	CART 10-fold	0.6378	82.69%	0.800	0.762	0.780
	KNN 10-fold	0.3707	69.23%	0.609	0.667	0.636
	SVM 10-fold	-0.0381	57.69%	0.000	0.000	0.000
All features except Delta	NB 10-fold	0.3196	63.46%	0.528	0.905	0.667
	LR 10-fold	0.1888	61.54%	0.526	0.476	0.500
	CART 10-fold	0.4686	75.00%	0.722	0.619	0.667
	KNN 10-fold	0.3963	71.15%	0.650	0.619	0.634
	SVM 10-fold	-0.0310	57.69%	0.000	0.000	0.000

All features except Percent Change	NB 10-fold	0.2281	57.69%	0.487	0.905	0.633
	LR 10-fold	0.1548	59.62%	0.500	0.476	0.488
	CART 10-fold	0.5573	78.85%	0.750	0.714	0.732
	KNN 10-fold	0.2353	63.46%	0.550	0.524	0.537
	SVM 10-fold	0.0000	59.62%	0.000	0.000	0.000

Table 5 – Feature ablation results table.

A2. Model Comparison

Model Type	Model Type	Kappa Statistic	Accuracy	Precision (wrt TRUE)	Recall (wrt TRUE)	F-Measure (wrt TRUE)
Naive Bayes	NB (on self)	0.3684	65.38%	0.538	1.000	0.700
	NB 5-fold Cross-Validation	0.3684	65.38%	0.538	1.000	0.700
	NB 10-fold Cross-Validation	0.3288	63.46%	0.526	0.952	0.678
	NB 20-fold Cross-Validation	0.2982	61.54%	0.513	0.952	0.667
Logistic Regression	LR (on self)	0.2012	61.54%	0.524	0.524	0.524
	LR 10-fold Cross-Validation	0.1548	59.62%	0.500	0.476	0.488
CART	CART (on self)	0.5769	78.85%	0.692	0.857	0.766
	CART 5-fold Cross-Validation	0.5706	78.85%	0.708	0.810	0.756
	CART 10-fold Cross-Validation	0.6006	80.77%	0.762	0.762	0.762
	CART 20-fold Cross-Validation	0.4321	73.08%	0.684	0.619	0.650
KNN	KNN (on self, K=1)	1	100.00%	1.000	1.000	1.000
	KNN 5-fold Cross-Validation	0.5207	76.92%	0.714	0.714	0.714
	KNN 10-fold Cross-Validation	0.4409	73.08%	0.667	0.667	0.667
	KNN 20-fold Cross-Validation	0.3963	71.15%	0.650	0.619	0.634
SVM	SVM (on self)	0.018	59.62%	0.500	0.048	0.087
	SVM 10-fold Cross-Validation	-0.0381	57.69%	0.000	0.000	0.000

Table 6 – table of training model evaluation results compared against different cross-fold validations.

A3. Testing model of model using new data sets

Test Set	Model Type	Kappa Statistic	Accuracy	Precision	Recall	F-Measure
Game 1	NB	0.1698	54.17%	0.441	0.833	0.577
	LR	0.2688	64.58%	0.524	0.611	0.564
	CART	0.2941	62.50%	0.500	0.833	0.625
	KNN	-0.0120	56.25%	0.364	0.222	0.276
	SVM	0.0000	62.50%	0.000	0.000	0.000
Game 2	NB	0.2799	63.04%	0.567	0.810	0.667
	LR	0.5551	78.26%	0.824	0.667	0.737
	CART	0.4283	71.74%	0.700	0.667	0.683
	KNN	0.4283	71.74%	0.700	0.667	0.683
	SVM	0.0082	54.35%	0.500	0.048	0.087
Game 3	NB	0.3684	65.38%	0.538	1.000	0.700
	LR	0.2012	61.54%	0.524	0.524	0.524
	CART	0.5769	78.85%	0.692	0.857	0.766
	KNN	1.0000	100.00%	1.000	1.000	1.000
	SVM	0.0180	59.62%	0.500	0.048	0.087
Game 1 (derived features)	NB	0.1402	52.08%	0.429	0.833	0.566
	LR	-0.0811	58.33%	0.000	0.000	0.000
	CART	0.2941	62.50%	0.500	0.833	0.625
	KNN	-0.0476	54.17%	0.333	0.222	0.267
	SVM	0.0000	62.50%	0.000	0.000	0.000
Game 2 (derived features)	NB	0.4048	69.57%	0.621	0.857	0.720
	LR	0.2164	63.04%	0.750	0.286	0.414

	CART	0.4283	71.74%	0.700	0.667	0.683
	KNN	0.2990	65.22%	0.619	0.619	0.619
	SVM	0.0082	54.35%	0.500	0.048	0.087
Game 3 (derived features)	NB	0.2281	57.69%	0.487	0.905	0.633
	LR	0.3456	71.15%	0.800	0.381	0.516
	CART	0.5769	78.85%	0.692	0.857	0.766
	KNN	1.0000	100.00%	1.000	1.000	1.000
	SVM	0.0000	59.62%	0.000	0.000	0.000

Table 7 – table of model results when tested against novel datasets.

Appendix B

Python Scripts

(see next page)

B1. Python Twitter Streaming Module for capturing live Twitter data

```
from datetime import datetime
import atexit
import json
import signal
import sys

from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
import MySQLdb
import _mysql_exceptions

access_token = "---"
access_token_secret = "---"
consumer_key = "---"
consumer_secret = "---"

TWITTER_FORMAT = '%a %b %d %H:%M:%S %z %Y'
MYSQL_FORMAT = '%Y-%m-%d %H:%M:%S'
INSERT_QUERY = """
    INSERT INTO tweets (
    tweet_id,
    created_at,
    text,
    hashtags,
    user_mentions,
    retweet_count,
    favorite_count)
    VALUES (%s, %s, %s, %s, %s, %s, %s)
"""

class TweetListener(StreamListener):
    def __init__(self):
        self.tweets = []
        self.counter = 1
        self.db = MySQLdb.connect(
            '---',
            '---',
            '---',
            '---',
            charset="utf8mb4",
            use_unicode=True)
        self.cursor = self.db.cursor()

        atexit.register(self.save_tweets)

        print('TweetListener is starting up...')

    def extract_entities(self, entity, key):
        return ';'.join([
            values[key] for values in self.data['entities'][entity]
        ])

    def on_data(self, raw_data):
        self.data = json.loads(raw_data)

        try:
            created_at = datetime.strptime(
                self.data['created_at'],
                TWITTER_FORMAT)
            tweet_id = self.data['id_str']
            tweet_text = self.data['text']
            hashtags = self.extract_entities('hashtags', 'text')
            user_mentions = self.extract_entities(
                'user_mentions',
                'screen_name')
            retweet_count = self.data['retweet_count']
            favorite_count = self.data['favorite_count']
        except KeyError:
            return True

        print(f'Found {tweet_id} tweeted at {created_at}')
        sys.stdout.flush()

        try:
            self.cursor.execute(INSERT_QUERY, (
                tweet_id, created_at.strftime(MYSQL_FORMAT), tweet_text,
                hashtags, user_mentions, retweet_count, favorite_count))
            self.db.commit()
        except _mysql_exceptions.OperationalError:
            print(f'Error on: {tweet_id}')
            pass

        self.tweets.append(raw_data)
        if len(self.tweets) >= 500:
            self.save_tweets()
        return True

    def save_tweets(self):
        print(f'Saving tweets in batch-{self.counter}.json')
        with open(f'rawdata/batch-{self.counter}.json', 'w') as output:
            json.dump(self.tweets, output)
        self.tweets = []
        self.counter += 1

    def on_error(self, status):
        print(status)
        if status == 420:
            self.save_tweets()
            return False

if __name__ == '__main__':
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)

    listener = TweetListener()
    twitterStream = Stream(auth=auth, listener=listener)

    twitterStream.filter(track=['#worlds2017', '@lolesports', '#ssgwin', '#sktwin',
'@sktelecom t1'])
```

B2. Python Script for calculating TF, TF.IDF for a given instance.

```
__author__ = 'Jenny Feng, jcfeng@live.unc.edu, Onyen = jcfeng'
import csv
import math
from textblob import TextBlob as tb

cloud = {}
bloblist = []


def tf(word, blob):
    return blob.words.count(word) / len(blob.words)

def n_containing(word, bloblist):
    return sum(1 for blob in bloblist if word in blob.words)

def idf(word, bloblist):
    return math.log(len(bloblist) / (1 + n_containing(word, bloblist)))

def tfidf(word, blob, bloblist):
    return tf(word, blob) * idf(word, bloblist)

with open('file.csv', 'r', encoding = 'UTF-8') as tweettext:
    reader = csv.reader(tweettext, delimiter=',')
    for row in reader:
        bloblist.append(tb(row[0]))
        words = row[0].split(' ')
        for word in words:
            if word in cloud:
                cloud[word] += 1
            else:
                cloud[word] = 1

with open ('freq.csv', 'w', encoding='UTF-8') as output:
    writer = csv.writer(output)
    for row in cloud.items():
        writer.writerow(row)


for i, blob in enumerate(bloblist):
    print("Top words in document {}".format(i + 1))
    scores = {word: tfidf(word, blob, bloblist) for word in blob.words}
    sorted_words = sorted(scores.items(), key=lambda x: x[1], reverse=True)
    for word, score in sorted_words[:3]:
        print("\tWord: {}, TF-IDF: {}".format(word, round(score, 5)))
```

Bibliography

2017 World Championship. (n.d.). From Leaguepedia: https://lol.gamepedia.com/2017_Season_World_Championship

Banerjee, M., Capozzoli, M., McSweeney, L., & Sinha, D. (1999). Beyond Kappa: A Review of Interrater Agreement Measures. *The Canadian Journal of Statistics.*, 27 (1): 3–23. doi:10.2307/3315487. JSTOR 3315487.

Hannon, J. M. (2011). Personalized and automatic social summarization of events in video. . *Paper presented at the 335-338. doi:10.1145/1943403.1943459*.

Lightside. (n.d.). From Lightside: http://ankara.lti.cs.cmu.edu/side/

Petrović, S. O. (2010). Streaming first story detection with application to twitter. *Paper presented at the 181-189*.

Petrović, S. O. (2010). Streaming first story detection with application to twitter. . *Paper presented at the 181-189.*

Powers, D. M. (2011). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies.*, 2 (1): 37–63.

Punera., D. C. (2011). Event Summarization using Tweets. *Paper presented at AAAI ICWSM.*

Sakaki, T. O. (2010). Earthquake shakes twitter users: Real-time event detection by social sensors. *Paper presented at the 851-860. doi:10.1145/1772690.1772777*.

Taylor, J. R. (1999). An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements. *University Science Books.*, 128–129. ISBN 0-935702-75-X.

TREC Real-Time Summarization Track Homepage. (n.d.). From http://trecrts.github.io

Tweepy. (n.d.). From Tweepy: http://www.tweepy.org/

Twitter Developer. (n.d.). From Twitter: https://developer.twitter.com/

Weka 3. (n.d.). From Weka - University of Waikata: https://www.cs.waikato.ac.nz/ml/weka/

Zhao, S. Z. (2011). Human as real-time sensors of social and physical events: A case study of twitter and sports games.

Zubiaga, A. S. (2012). Towards real-time summarization of scheduled events from twitter streams.

Martínez, P., Segura, I., Declerck, T., & Martínez, J. L. (2014). TrendMiner: Large-scale cross-lingual trend mining summarization of real-time media streams. Procesamiento De Lenguaje Natural, 53, 163-166.

Liu, C., Chen, M., & Tseng, C. (2015). IncreSTS: Towards real-time incremental short text summarization on comment streams from social network services. Ieee Transactions on Knowledge and Data Engineering, 27(11), 2986-3000. doi:10.1109/TKDE.2015.2405553

Chellal, A., Boughanem, M., & Dousset, B. (2017). Multi-criterion real time tweet summarization based upon adaptive threshold. Paper presented at the 264-271. doi:10.1109/WI.2016.0045

Kedzie, C., Diaz, F., & McKeown, K. (2016). Real-time web scale event summarization using sequential decision making.

Zhou, Y., Kanhabua, N., & Cristea, A. I. (2016). Real-time timeline summarisation for high-impact events

in twitter. Paper presented at the , 285 1158-1166. doi:10.3233/978-1-61499-672-9-1158

Ekstrand-Abueg, M., McCreadie, R., Pavlu, V., & Diaz, F. (2016). A study of realtime summarization

metrics. Paper presented at the , 24-28- 2125-2130. doi:10.1145/2983323.2983653

Khan, M. A. H., Bollegala, D., Liu, G., & Sezaki, K. (2013). Multi-tweet summarization of real-time events.

Paper presented at the 128-133. doi:10.1109/SocialCom.2013.26

Ren, J., Jiang, J., & Eckes, C. (2008). Hierarchical modeling and adaptive clustering for real-time

summarization of rush videos in trecvid'08. Paper presented at the 26-30.

doi:10.1145/1463563.1463566

Roegiest, A., Tan, L., Lin, J., & Clarke, C. (2016). A platform for streaming push notifications to mobile

assessors. Paper presented at the 1077-1080. doi:10.1145/2911451.2911463

Kubo, M., Sasano, R., Takamura, H., & Okumura, M. (2013). Generating live sports updates from twitter

by finding good reporters. Paper presented at the , 1 527-534. doi:10.1109/WI-IAT.2013.74

Esmin, A. A. A., Júnior, R. S. C., Santos, W. S., Botaro, C. O., & Nobre, T. P. (2014). Real-time

summarization of scheduled soccer games from twitter stream. Paper presented at the , 8455 220-223. doi:10.1007/978-3-319-07983-7_29

Nichols, J., Mahmud, J., & Drews, C. (2012). Summarizing sporting events using twitter. Paper presented

at the 189-198. doi:10.1145/2166966.2166999

Anjum, M. E., Ali, S. F., Hassan, M. T., & Adnan, M. (2013). Video summarization: Sports highlights

generation. Paper presented at the 142-147. doi:10.1109/INMIC.2013.6731340