NHL Event Detection

Bofei Jing, Brendan Artley, Shrey Grover, Kukpyo Han

Master of Science in Professional Computer Science - Big Data

Simon Fraser University

Dec 10th, 2021

1. Project Overview

The project aims to determine how well human sensors can detect high frequency less significant (HFLS) events such as those occurring during a nationally broadcasted sports tournament like National Hockey League (NHL). This project envisions providing a proof-of-concept to detect more pressing HFLS events such as plea calls during natural disasters or traffic accidents in real-time.

The problem at hand is occupied with two major challenges -

(1) to fetch and associate relevant tweets (or comments) to respective games from a vast pool of available microblogging data
(2) to process high-frequency incoming tweets (or comments) without significant delay

The approach adopted to address the problem leverages key sensing properties of human sensors like post rate and incorporates adaptive sliding windows to mitigate detection delays. This project is limited to detecting two major buckets of events - goals and not goals due to the high occurrence of these events in a hockey game.

2. Data Used

The dataset for this problem consists of two major aspects - tweets (or comments) collected from microblogging sites (Twitter and Reddit) and ground truth game event data from NHL games, each of which was collected across two seasons (2019-20 & 2020-21).

Sources include

- Reddit Corpus Dataset (~500GB)
- Scraped Twitter Data (~50GB)
- NHL API (~2GB)

3. High-level System Architecture

Fig 1 shows the high-level system architecture. A detailed explanation of each step is provided in the following sections.

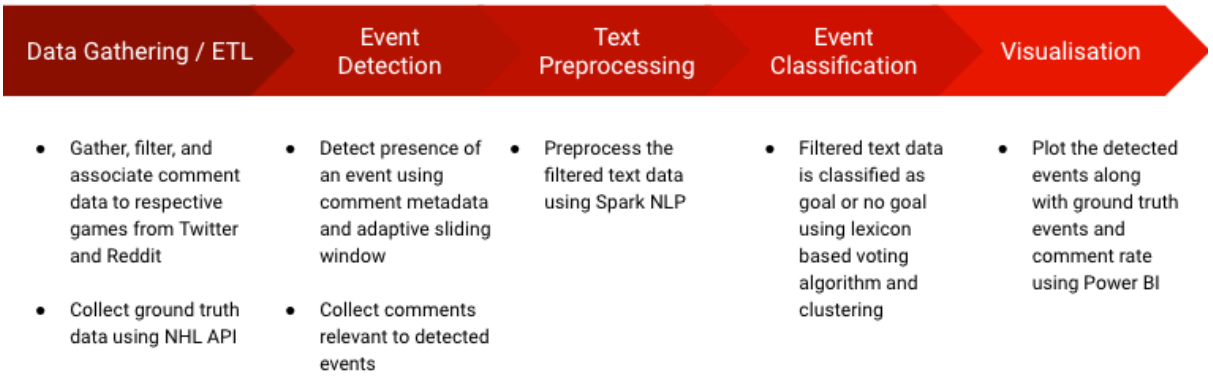

Fig 1: Project Steps

4. Implementation and Methodology

4.1 Data Gathering

4.1.1 NHL API

The NHL provides a very exhaustive API to the public. The project integrated this ground truth data source to help with data collection and prediction evaluation. Among the various endpoints, three APIs were most heavily used, including the APIs for team information, game schedule, and historical or real-time in-game events. A scraper was developed to retrieve data from the API. As there are few limitations imposed by NHL on the request frequency or maximum parallel requests, this script employs asynchronous programming techniques to gather all the data as fast as possible and execute in a matter of minutes. Subsequently, the data was flattened from its JSON format, trimmed of unnecessary fields, and finally transformed into parquet file format for future use.

4.1.2 Reddit Data

Since there are no readily available NHL comment datasets, this project utilized Spark to filter relevant comments that occurred during NHL games from the entire Reddit comments corpus. Firstly, this involved filtering through the submissions (aka posts) by the 32 individual team subreddits and the “hockey” subreddit. Next, each subreddit classified game threads differently. Using filtering combinations of “link_flair_text”, “title”, “selftext” and in some cases manually through the “id” field the game threads were extracted for each team. Exploring the filtering methods that were relevant to each subreddit was done through the Spark shell and Ipython.

At this point, the Reddit submissions were ready to be joined with the NHL API data. The NHL API data contained information on all the events that occurred during each game, the home team, the away team, unique ids, and more. It was paramount to accurately join this game information with each submission. Since many games happened at the same time, it required engineering multiple additional columns on the Reddit data. For each game, there were between 1-3 game threads found in different subreddits. One for the home team, one for the away team, and one “neutral” comment thread in the “hockey” subreddit.

Furthermore, in order to accurately complete the join string parsing was used to retrieve the home team and away team from the “hockey” subreddit titles, and when/otherwise clauses were used to retrieve the team name from the individual teams subreddit and post title. Then converting the timestamps to dates, the day was extracted from the timestamp and used to join the nhl_api data with the comments. Using the events game_start and game_end time, comments were filtered out that were made before or after the game. Finally, feature columns were added that indicated if a goal or penalty had occurred in the 60 seconds after each comment.

Given the different nature of the Reddit data and NHL API data, the ETL process was technically challenging and required the use of a variety of data frame operations, pyspark.sql.functions, UDF's, and Datatypes in order to unify the datasets. Much of this work was split into multiple scripts, and chaining these together was made easier through the use of the parquet file format. The ETL pipeline was built to scale in order to maintain efficiency as the size of the dataset increases.

4.1.3 Twitter Data

Since there is no readily available NHL-related Twitter data, the data in this project was generated by scraping tweets using an open-source third-party tool called Twint. The scraping process involved looking for tweets that were most likely to be related to NHL games. This involved using certain keywords, hockey-related hash-tags, and game times from the NHL API. These keywords included the full name of NHL teams, abbreviated team names, and the keywords "NHL" and "hockey". This small list of keywords was used mostly due to the fact that, unlike Reddit, Twitter as a social media platform is not subdivided into "Sub Twitters". As a result, when the searching criterion was relaxed, too many unrelated tweets would appear in the raw data, making the dataset full of irrelevant tweets.

In the scraping tool Twint, asynchronous programming techniques had already been applied by the library author to speed up request sending and processing. That being said, to make gathering all game-related tweets in the past three years practical, multithreaded programming was also applied on top of the tool itself. This enabled the collection of more than twenty million tweets in a matter of hours.

The ETL process for tweets was relatively straightforward. Due to the fact that the scraping tool supported only day-leveled DateTime filtering and no language filtering, these constraints had to be implemented in the ETL process. This required joining the dataset against the game schedule data gathered from the NHL official API. Other than that, the data was highly structured and ready to use and therefore required little transformation after this step.

4.2 Event Detection

Once the relevant data had been gathered, the next task was to detect the presence of events throughout each game. This detection method was built using the metadata of the tweets (or comments) and the concept of adaptive windowing. Adaptive windowing allows the system to mitigate detection delays caused by a large fixed window. The predefined sub-windows are either 6, 10, 20, 30, or 60 seconds long. The following logic was implemented to detect events:

1. The tweets (or comments) were arranged in chronological order and bucketed into batches of the predefined sub-window size.
2. For each sub-window (starting from the smallest), the following conditions were verified for every batch:
    a. The post-rate (i.e. number of tweets (or comments) per second) of the sub-window should be greater than 1
    b. The post-rate of the second half of the sub-window should be greater than the post-rate of the first half
3. If any of the above conditions are not met, a bigger sub-window is considered. This process is repeated until either the conditions are met or all the predefined sub-windows are exhausted.
4. If a sub-window is identified that satisfies the above conditions, an event is said to have been detected in this window.
5. To identify the tweets (or comments) related to the detected event, all tweets (or comments) occurring after the mid-point of the sub-window are obtained.
6. The timestamp of the midpoint of the sub-window is considered as the time of the detected event.

Implementing this strategy in Spark in a scalable fashion was difficult and required pursuing many different methods. The method that yielded optimal performance was through a Pandas UDF. This UDF directly assigned each comment (or tweet) to all the windows it belonged to. After acquiring this information rows were aggregated by these unique window identifiers. Using this method, unnecessary windows that contained

no comments could be avoided. This reduced the amount of data in memory at one time and was the most efficient method in detecting events.

4.3 Text Cleaning

The text cleaning involved using a package called spark-NLP. This library is built on top of Spark and its built-in ML library. Using spark-NLP's pipeline tools comments were able to be processed quickly and efficiently. The pipeline consisted of a Tokenizer, Normalizer, Lemmatizer, and Stop Words Remover. The tokenizer splits each comment by space. The normalizer sets all the words to lowercase and removes unnecessary punctuation through regex pattern matching. The Lemmatizer is a trained component that provides spelling correction based on common English language errors. Finally, the stop words remover removes stop words. Some examples of these words include “a”, “the”, “is”, or “are”.

Furthermore, using Spark, words that had the text “[removed]” or “[deleted]” were filtered after the event detection process. The comments with these string values were important in determining the activity of users throughout each game but did not convey any important information that could be used for classification.

4.4 Event Classification

For the event classification component, a lexicon-based method was used to determine whether each detected event was a goal or not. The comments in each window were compared with a collection of lexicons generated from word frequency counts, TF-IDF scores, and common NHL terminology. If the average number of comments in a given window contained these lexicons, then the mid-point of that window was classified as the time of a goal. If the average number of lexicons did not reach that threshold then this event was determined not to be a goal.

The next part of the classification was to perform clustering on the windows that predicted goals. Without this step, there would be many windows predicting the same event. To implement this clustering algorithm, all the predicted goals in an event were aggregated using pyspark.sql’s collect list function. Once aggregated, a UDF was applied to determine if each window was within 30 seconds of its nearest neighbors. If this was the case, these windows were clustered together. Although a UDF was added, the number of predictions that were filtered out resulted in a decrease in the overall runtime of the application.

Furthermore, the implementation required using varying window sizes between 6-60 seconds. Through trial and error, it was determined that any window size below 20 seconds did not produce enough comments in the window to accurately classify events. Given more comment data, it may have been beneficial to decrease the window size, but in this case, 30-60 seconds was the optimal range.

4.5 Visualization

For visualizations in the presentation, a combination of plots created in Tableau, Draw.io, and Power BI were used. The majority of plots were created using Tableau Desktop. These plots were helpful in the exploratory analysis, and for visualizing findings and results. The activity of each fan base and the total number of fans' comments were visualized using bar plots. Word clouds were used to visualize the most frequently appearing words in comments after goals, and for all other comments. Finally, line graphs were used to show comment activity over time, the time of goals during the games, and the predicted times of goals. Draw.io was also used to visualize components of the sliding window method and clustering algorithms. These tools aided in translating complex techniques into more comprehensible graphics.

Finally, using Power Bi’s interactive Pulse Chart, the predicted goals were compared with the true labels for the final part of the presentation.

5. Challenges Faced

The implementation posed certain challenges, mainly stemming from the Data Gathering, ETL, event detection, and classification steps. Some of these challenges are highlighted below:

1. Associating comments to the appropriate games due to the diversity of subreddit formats in Reddit game threads. This required string parsing on the game thread title, to retrieve team names, and game date to accurately identify the equivalent game.
2. Obtaining sliding batches of chronologically ordered comment data while maintaining scalability. It was made possible by using Pyarrow and a user-defined function. Other solutions would require materializing all windows for each second during a game, and joining them back to the comments. This would result in unmanageable application run times.
3. The sparsity of comments in the data forced some changes in the detected event conditions, which required experimenting with a range of parameter values.
4. Dealing with timestamps was also a major hurdle as Spark converted the timestamps to local time, which motivated the use of UNIX timestamps to maintain homogeneity.

6. Future Work

There were a few aspects of the implementation that could be improved on. First of all, the lexicon-based classification method was relatively low in complexity compared with most modern classification methods. To improve the accuracy of classifications it could be beneficial to explore using state-of-the-art NLP technologies. For example, when a goal is scored it is likely that the game thread for fans on each team is heavily skewed in its sentiment. This information could potentially increase the predictive accuracy of the model.

Game threads are also relatively new components of social media sites, only being adopted by all the NHL team fan bases in late 2020. Although these threads are quite new, they are quickly increasing in popularity. A large portion of the games prior to 2021 did not have enough comments made during the games to accurately detect events and classify goals. Increasing the number of comments through more effective filtering methods, or additional social media sites could mitigate this issue.

7. Results

The precision accuracy of the model was 72% on the Reddit Data, and 80% on the Twitter Dataset. A prediction was marked as a true positive if the predicted goal time fell within 60 seconds of a true label. The second metric used was the overall recall accuracy. This measured the percentage of all of the true labels that were accurately predicted. The recall accuracy of the model on the Reddit data was 50%, and 36% on the Twitter dataset.

Given the sparsity of some game threads, it was expected that the recall accuracy would not be as high as the precision accuracy. That being said, there were many games that had great predictions throughout the match. For example, in figure 2 the predictions made by the model are compared with the true labels of goals during the match. Note that the green marks correspond to the true labels and the red marks correspond to our predictions.

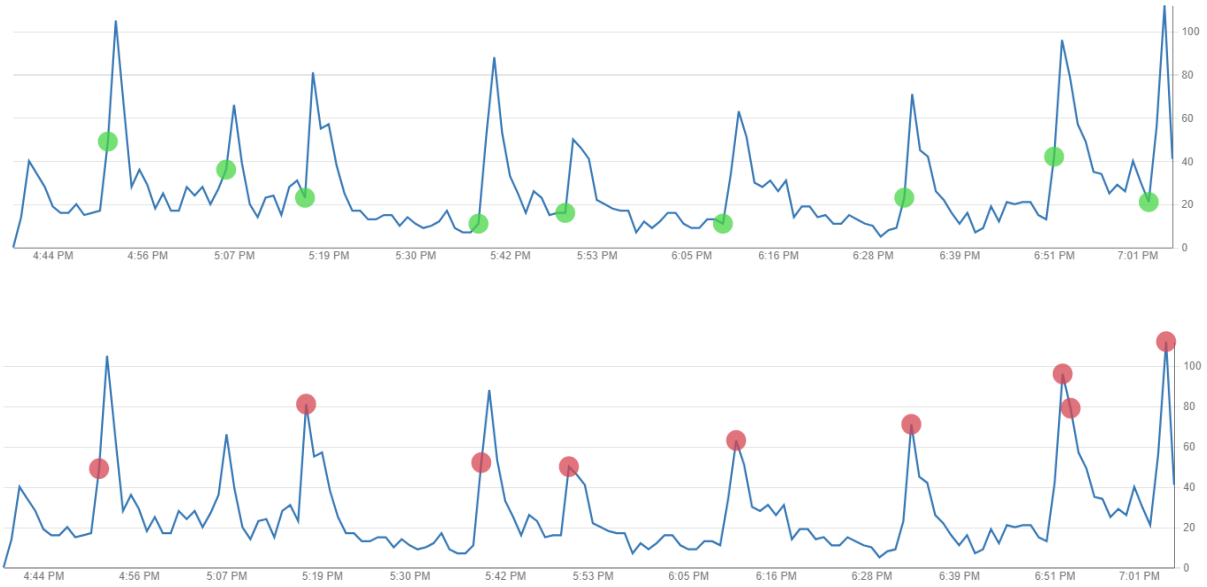

Fig 2: NHL Game Timeline - Event Detection

8. Project Summary

Category	Description	Score
Getting the data	Acquiring/gathering/downloading.	5
ETL	Extract-Transform-Load work and cleaning the data set.	7
Problem	Work on defining problem itself and motivation for the analysis.	0
Algorithmic Work	Work on the algorithms needed to work with the data, including integrating data mining and machine learning techniques.	3
Bigness/ Parallelization	Efficiency of the analysis on a cluster, and scalability to larger data sets.	4
UI	User interface to the results, possibly including web or data exploration frontends.	0
Visualization	Visualization of analysis results.	0
Technologies	New technologies learned as part of doing the project.	1