Fantasy Sports News - A RESTful Web Application Using Yahoo API

C. MacQuarrie and J. Rizzo

April 9, 2017

1 Introduction

1.1 Context

The Fantasy Sports is a 2.6 Billion dollar industry with over 58 Million people participating in 2016.[1] With players averaging three teams and each team having aproximately 20 players the time investment to keep an upper edge is significant.

1.2 Problem Statement

With 750 players currently active in the MLB, 450 more under contract, and thousands more in development leagues, maintaining information on all of them is near impossible. There are hundreds of sources of information available but sifting through the noise can be overwhelming. Players will either invest a significant amount of time researching or lose the edge to an opponent who does.

1.3 Results

Fantasy Sports News (FSN) is a $RESTful$ web service utilizing Jersey. Utilizing the Yahoo Fantasy Aplication Program Interface (API) all player, team, and user information is collected automatically. FSN constantly crawls known, trusted webpages for new content. Storing these documents in a *Lucene* index allows quick queries for specific players. Users can filter news based on owned players, free agents, or specific players.

1.4 Outline

The report is laid out as follows. Section 2 provides background information prudent to the report, including but not limited to; Fantasy Baseball, RESTful APIs, Yahoo Fantasy API, Indexing, and Sentiment Analysis. Section 3 covers

[1] `https://www.stats.com/industry-analysis-articles/fantasy-sports-industry-demographic/`

the implementation of FSN and the results observed. In section 4 we discuss the benefits of FSN. In section 5 we conclude the report and propose future work to be done.

2 Background Information

2.1 Fantasy Baseball

Fantasy Baseball is a competition between people, often with money wagered. Competitions range from single day events to yearly leagues or longer. Competitors draft teams of MLB baseball players whose real performance scores for the competitors fantasy team. Generally only one competitor can own a single MLB player. Throughout the course of the competition competitors can trade players, add free agents, and drop players.

2.2 RESTful APIs

RESTful services allow Create, Read, Update, Delete (CRUD) functions through HTTP requests. Get Requests are used to retrieve resources, Put requests are used to create resources, Post requests are used to update resources and Delete requests to delete[2]

2.3 Yahoo Fantasy API

Yahoo provides a convenient API used for the retrieval of the data in their fantasy sports system. This API makes it possible for developers to pull resources such as leagues, teams, and players of a particular Yahoo Fantasy Sports user. It works on a RESTful model, so data is pulled via HTTP requests, with a typical request URLs being formatted as follows:

```
http://fantasysports.yahooapis.com/fantasy/v2/{resource}/{resource_key}

http://fantasysports.yahooapis.com/fantasy/v2/{collection};{resource}_
keys={resource_key1},{resource_key2}
```

In order to use Yahoos Fantasy Sports API, the application needs the users permission to make requests on their behalf. Yahoo has implemented this functionality using a system called OAuth. OAuth gives the user the ability to grant access to their private data on a service provider (in this case the service provider is Yahoo Fantasy Sports), to another site called the consumer (our application). From the time the consumer application is first opened, to the time the application is granted access to the service providers data by the user, the following workflow is executed:

1. A request token is requested by the consumer application using its unique API key and secret

[2] `http://stackoverflow.com/questions/630453/put-vs-post-in-rest`

2. When the request token is received, it is used to request an access URL
3. The user is then redirected to this URL, where they will sign in to their account and then click a button to grant access
4. The user is redirected back to the consumer site using a pre configured callback URL
5. A verification code is included in the callback URL as a parameter which is used along with the request token to request an access token
6. The consumer site now has access to the users private data on the service providers site, and every request to the site must signed with the access token

2.4 Indexing

In order to quickly search the documents for terms you must index them. This creates a dictionary of terms and associates documents with them. Apache Lucene is a java library that can be used to index and search documents.

2.5 Sentiment Analysis

Sentiment Analysis was used to evaluate the articles obtained while crawling the web in this project. We want to be able to present the articles in such a way that the most relevant articles are seen first, and sentiment scoring can help with this. An extreme sentiment score in both a positive or negative direction likely indicates an article with subject matter that is more pressing than an article with a neutral score. For this project, sentiment score of each sentence was obtained using the Stanford CoreNLP library. If the sentiment was very negative it got a -2, negative a -1, neutral a 0, positive a 1, and very positive a 2. The average was then taken to get an overall sentiment score for the article

3 Results

3.1 Data Structures

- A *User* is created when a login occurs. Users have an id, teams, and OAuth Token
- A *Team* is created when a login occurs. Teams have a name, id, league key, and Players.
- A *Player* is created when seeding occurs. Players have a first name, last name, id, and rank.
- An *Article* is created during crawling. Articles have a title, id, url, publish date, sentiment value, score and source.

3.2 Implemented Program Control Flow

3.2.1 Web Services

The following web services we implemented:

- GET / - Link to allow user to log in using Yahoo
- GET /retrieveplayers - Used to seed all player data from Yahoo
- GET /players - Displays a list of links to all players profiles
- GET /players/id - Displays information and articles about a player
- PUT /players/id - Creates or updates a player
- GET /teams - Displays a list of all current users
- GET /teams/id - Displays all players and articles relevant to a team
- GET /teams/id/freeagents - Show top free agents and relevant articles
- GET /news - Displays all article
- GET /news/id - Displays the details of an article
- POST /news/id - Creates or updates an article
- GET /news/search/query - Queries Lucene for relevant articles. Returns as HTML
- GET /news/search/query - Queries Lucene for relevant articles. Returns as JSON

3.2.2 Gathering

A list of article index location was created. From this list, we used the *Jsoup* HTML parsing library to gather links to articles. In order to initially seed our database with past articles, we set our last crawl to the beginning of the year. Subsequent crawls will only pull articles up to the time of the previous call. For the testing purposes of FSN, the websites crawled were;

`http://m.mlb.com/news/`

`www.reddit.com/r/fantasybaseball`

`www.espn.com/baseball`

3.2.3 Storing

Since we gathered a large quantity of data with relatively little relationships between them we used MongoDB. We stored our permanent data in two collections, players and articles. We did not chose to store user information as we could not guarantee the security of our software.

3.2.4 Analyzing

The sentiment of the articles was analyzed as the documents were crawled. We used *CoreNLP* as the library to do this. Every line was analyzed on a 5-point scale. The articles sentiment was decided to be the average sentiment of all lines it contains. Articles were also fed to Lucene to be indexed. Stopwords were removed and terms were assigned to articles.

3.2.5 Filtering

Lucene was used to generate queries of our database. The primary filtering we are concerned with is based on players last names. By separating arguments in the query with a space (' '), 'or' statements can be created. Since web browsers in general don't support spaces in URLs, we sanitized our queries by replacing "%20" with a space. Multiple players can then be queried and all relevant results will be returned.

3.2.6 Displaying

FSN uses HTML to display its services. The standard page layout is two pages; player information on the left and relevant articles on the right. When a user logs in they are directed to a page with links to all their teams. An HTML rendering system was created to ensure consistently formatted pages.

3.3 Observations

One of the biggest things we noticed is that all of the articles sentiment scores, especially ones from trusted providers, had very similar sentiment scores, which ranged from around 0.5 0.7. We believe that this is because journalists try their best to unbiased and use relatively neutral and precise terms. The exception to this rule were the posts grabbed from Reddits subreddit r/fantasybaseball. This is likely because these posts were not written by journalists, and were on average a lot shorter than professionally written articles. Therefore there is more of a tendency to be biased and exaggerate, and there are not a lot of sentences to even out the averaged sentiment score.

Another minor thing we noticed is that some website layouts were rather arbitrarily designed and were difficult to parse and extract the exact data that we needed to store. This can be problematic because if we have any data stored from a page that we did not intend to store, such as advertisement content, or text from another article on the page, or any other unrelated data, it could affect the search results. This can also be true if we left out data that we wanted to store.

4 Evaluation

4.1 What was Implemented

We were successful in creating a consolidated location for getting news. A RESTful service was created that allows for expanded interactions with the database. OAuth was implemented allowing the ability to use Yahoo Fantasy API. Searching quickly is possible with the integration of Lucene. Sentiment of articles was analyzed and stored. Crawlers were maintained and resources actively collected.

4.2 Difficulties Faced

Due to the small variance in sentiment we were unable to implement events. Extracting information about which players were the focus of which articles also proved to be an difficulty we could not overcome. We proposed using Twitter feeds as a source of data, however due to the frequency of our crawl schedule was not implemented. As tweets are limited to 140 characters for the information gathered from Twitter to be valuable it must be consumed very quickly after being published.

5 Conclusion

5.1 Summary

FSN allows users to visit a single location to consume all their fantasy news. Utilizing Yahoo's API minimal work is needed on the users end to maintain the news pool. Allowing users to filter news saves research time and increases research efficiency.

5.2 Future Work

There are three ways to expand FSN; expanding our platform to feature additional sports, allow additional platforms beyond Yahoo, and increase the functionality of the API calls. Our implementation is not tied to any aspects of Baseball so adding more sports would be possible. The first sport to expand to would be Football as it has the largest population of players. Implementing additional platforms outside of Yahoo would increase the user base in the long run. Since FSN is currently no distributed this is of low priority. Additional features could be added through Yahoo's API. Giving the users the ability manipulate their players and rosters using FSN would be highly beneficial.

A References

1. *Fantasy Sports Industry Demographics*, 2017, URL:`https://www.stats.com/industry-analysis-articles/fantasy-sports-industry-demographic/`