Taras Tataryn  
2/09/18

Proposal for

**EPL Transfer Aggregator**

[I. Summary 3](#_Toc52873031)

[II. Significance](#_II.__Significance) 4

[III. Required Tools and Availability](#_III.__Required) 4

[IV. Demonstration Plans](#_IV.__Demonstration) 5

[V. Qualifications](#_V.__Qualifications) 5

[VI. Project Specification](#_VI.__Project) 6

[VIi. tECHNICAL DETAILS](#_VII.__Technical) 7

[VIIi. Timeline](#_VII.__Timeline) 9

[iv. Future enhancements](#_VII.__Future) 12

[x. Bibliography](#_Toc52873040) 12

# Summary

# *EPL Transfer Aggregator* is a web application that tracks EPL (English Premier League) soccer transfer rumors through Twitter. The overall goal is to provide users with a regularly updated page of transfer rumors for an EPL teams. However, with the abundance of tweets sent out daily by British reporters and newspapers, many of these rumors can be false. The *EPL Transfer Aggregator,* based on historical accuracy of reporters’ tweets and a tier system, will have a Transfer Reliability Guide to help users decide which rumors and reporters to trust. The user can then select a team and see the latest tweets and rumors from a select list of reporters to better see who is reporting on what. Finally, the web application will also provide graphs and statistics to better visualize transfer rumors, reports, etc. in the world of Twitter.

# II. Significance

The scope of this project fits well within and includes concepts learned in the Web Programming and Cyber Security course, as well as utilizes content learned from the Human and Computer Interactions course at DePauw. First, in CSC340 (Web Programming and Cyber Security), HTML5 and CSS were covered to help students become familiar with and have a basic understanding of how to create a simple webpage. These skills were refined using Ruby, HTML5, CSS, and learning Ruby on Rails, an important supplement to Ruby that employs a full-stack web-application framework that is written in Ruby. HTML and CSS will both be used in development of the website. Additionally, concepts learned in the Human Computer Interaction course will aid in creating a well-designed user interface. This includes fundamental principles such as avoiding cognitive overload, user error prevention, user-centered design, etc.

# III. Required Tools and Availability

This project is planned to be developed using the Sublime text editor with Python 3 on my Mac machine. The tools and Python packages I use are installed using the Pip installer via terminal. Other languages I utilize are HTML5, CSS, and some Javascript.

# IV. Demonstration Plans

Demonstration will take place in the Julian Center for Science and Math,

Room 260. The code, visuals, and/or presentations will be available and presented from my Mac computer. In order to navigate to the website on those specific checkpoints, I will access the web application via terminal and local host through Flask, which will display the web application. Additionally, In the case of connecting to a projector, I will need an HDMI cord to project my laptop to the screen as I have a Macbook Pro.

# V. Qualifications

EPL Transfer Aggregator is developed with HTML5, CSS, Python, and Flask, a micro web-framework for Python. I have completed multiple projects and on my own where I have created a website using HTML and CSS, and refined those skills in CSC 340: Web Programming and Cyber Security, which qualify me to develop this website as a senior project. I learned python on my own, but also brushed up on it and learned a lot more in my past internship experiences. In my internship last summer, one of my tasks was to create a locally hosted website with a database back-end using the Flask micro-framework. The web-scraping portion of my project was learned on my own as well, and utilized for a project started in the Data Science club. Finally, completing the CSC 320: Human Computer Interactions course, as aforementioned, will be important in implementing and using good design principles for my website.

# VI. Project Specification

*EPL Transfer Aggregator* intends to compile soccer transfer rumors that are sent out via Twitter. During peak transfer season (before the summer and winter transfer windows), English, European reporters, and transfer rumor accounts send out thousands of tweets daily. *EPL Transfer Aggregator* will collect these rumors and sort them by team/reporter as well as by who is the most accurate. My motivation for this project comes from being a big EPL (English Premier League) soccer fan. In the summer and winter, I am always on Twitter to check and see the latest transfer rumors. However, some of the tweets and rumors are very inaccurate and never turn out to anything. In fact, some Twitter accounts and reporters will just tweet out something random to get more “likes”, “retweets”, or followers. Thus, the goal of my website/project is to collect rumors from various reporters and display it in a format that is easy to locate trustworthy sources and rumors.

As a result of this, the intention is to have an elegant yet simple user interface where users can take a look at the Transfer Reliability Guide to understand and decide which sources are the most trustworthy. They then are able to access the Rumor Aggregator page, select a team from the dropdown menu and look at the latest rumors.

# VII. Technical Details

In order to create *EPL Transfer Aggregator,* it’s first necessary to design, implement, and finalize the template or blueprint of the web-application. As is typical, the front-end design will be written in HTML5 and CSS in the Sublime Text Editor. Using these two in conjunction with Flask will allow me to constantly see how my web-application is responding to the changes in my code and progress I am making. As I previously mentioned, Flask is a Python web micro-framework that allows for easy templating, a development server and debugger, very extensive documentation, and plenty of other features making it easy to host and create a web application. A web browser will be needed to actually see the progress of my development.

Since Flask depends on some external libraries for development and deployment, *virtualenv* is needed to get this on my computer quickly. *Virtualenv* is a tool that allows you to create isolated Python environments (a virtual environment), and will enable multiple side-by-side installations of Python. It’s simply installed and activated via command-line and pip, and that is all that is needed.

In terms of the file structure within the flask environment, I will have my starting Python file, for instance *app.py (*which will start the development server and contain other Python code), and then within that folder I will have a **static** folder. This contains the public CSS, Javascript, and images used to style my application. Additionally, I’ll have a **templates** folder, containing the html files that actually will create my webpages.

With regard to collection of data, I plan to use Tweepy to collect data and store into a CSV file and then analyze it from there. Using the Python *csv* module, I can easily write and read into a CSV file from my code, and use it to gather data such as *Player name, Team name,* etc. and pull it into a Pandas DataFrame. A DataFrame is essentially a 2-D labeled data structure that acts like a spreadsheet or SQL table. It’s a crucial data structure for my web application and will be used to display the transfer rumors, reporting accuracy, and more. Finally, the statistics, and graphs/visuals will be created using *matplotlib* (a Python 2D plotting library) in conjunction with the Panda library.

# VIII. Timeline

**Checkpoint 1:**

* I will have a compiled list of EPL transfers the last year or two with player name, team, and date of transfer (for historical use) which can be viewed in an Excel file. This is accomplished via Twitter’s “Advanced Search” function, and pulled into Excel. This data will be used to compute the historical accuracy presented in *Checkpoint 2.*
* I will decide on a list of reporters that I will be gathering data from.
* I will demonstrate sample tweets for a specific team pulled from Twitter and read into a CSV file with tweet text, username, and link to the tweet. The specific team will be chosen via user input on the web application (a dropdown menu). This will be presented via the Sublime Text Editor.

**Checkpoint 2:**

* I will have the frame of my website built (i.e. the navigation panels, individual pages built – blank -- but with the navigation panels available). This will be completed via a flask application (a Python micro-framework) and using HTML5 and CSS to style the pages.
  + Pages will include: a **Home** page with a preview and description about what the website/project is about, a **Transfer Reliability Guide** page with information on which sources to trust, a **Newspaper/Reporter Transfer Statistics** page with the statistical graphs and a visualization included, and the **Rumor Aggregator** page which actually has the rumors by team and reporter/newspaper.
  + As the Home page is separate from the rest of the future items in these Checkpoint, I will show the finished home page with all its content. The rest of the pages’ content will be displayed as more of the checkpoint items are completed.

I will have how accurate a newspaper has been for a specific period of time – i.e. The Times was 40% correct in the summer of 2017. This will be demonstrated in an Excel file. Tweets will be retrieved programmatically using the Python *GetOldTweets* package since Twitter’s API does not allow for searching past tweets. This will be presented in the form of a *Pandas dataframe* on the **Newspaper/Reporter Transfer Statistics Page**.

* I will have compiled a list of who (what newspaper or reporter) broke what transfer first. This will be accomplished using the list of transfers from *Checkpoint 1* and found using the *GetOldTweets* package. It will also be presented on the **Newspaper/Reporter Transfer Statistics** page on the web application.

**Checkpoint 3:**

* I will have created an algorithm that will be able to go through tweets and find those that are transfer rumors/news rather than just team news. This will be done using Named Entity Recognition in conjunction with Python’s NLTK package. The algorithm will gather tweet text from a CSV file and determine if there is a player name and team in the tweet.
  + The data will then be used to put into a Dataframe with the player and team name, source (reporter or newspaper), and link to tweet. This will be exhibited through my Python code and the CSV file.
* I will create and display graphs and a visualization created via the Python Pandas package. They will include: a histogram showing the number of transfers over time, rumors/tweets per day by reporter, and rumor accuracy by newspaper. The graphs will be exhibited on the **Newspaper/Transfer Statistics page.**

**Checkpoint 4:**

* The **Transfer Reliability Guide** page will be completed and presented. Based on the computed historical accuracy by reporter/newspaper and research, the guide will contain Hierarchal Tiers (1-3) with the corresponding reporters and newspapers. For instance, Tier 1 will have John Smith from Sky Sports, meaning he is very accurate and reliable.
* The **Rumor Aggregator Page** will include the dropdown menu – giving the user the ability to select a team and the program will gather a list of recent rumors from reporters and newspapers and display them in a dataframe as previously mentioned (Source, Player Name, Tweet Text and Link to Tweet).
  + Additionally, to personalize the page, a banner/club logo of that team will appear below the rumors.

# IV. Future Enhancements

* Implement in the program the ability to gather live tweets from Twitter and filter by looking at verified sources rather than looking only at a list of reporters that I created.
* Produce more meaningful visualizations (as I have only a little experience in this field, more education and practical experience will help).
* Look for rumors in teams outside the English Premier League – the Premier League is always at the top for amount spent per transfer window, but other leagues have an abundance of transfer rumors as well.
* Produce a more powerful and accurate algorithm for name extraction from Tweet text – it can be extremely difficult as not every Tweet from a reporter will be identical.

# X. Bibliography

# <https://github.com/Jefferson-Henrique/GetOldTweets-python>

# <http://docs.tweepy.org/en/v3.5.0/>

# <http://flask.pocoo.org/>