Extraction of Sports Particulars as Networks (ESPN) API/CLI

Collin J. Kovacs
Indiana University
cokovacs@iu.edu

I. Abstract

This implementation project, Extraction of Sports Particulars as Networks (ESPN), will allow users to study sports analytics in a unique way. Most sports data is given in either a tabular format, or as an unstructured set like a JSON response object from an API. For networks to be created, specific nodes of sources and targets with edges need to be specified within the dataset being used. To bypass this stringency, ESPN will allow users to query certain parameters like type of sport, team or player, and receive their data in a network like format like Graph Markup Language (GML). This allows for understanding underlying connections at the team or player level.

II. Introduction

This project is the initial approach towards reframing data written in tabular or unstructured data formats to produce meaningful and novel insight through network analysis. It is common for loading tabular data like Comma Separated Values (.csv) format that explicitly contain the nodes (source and target) and edges or where edges are inferred through parsing through the nodes. From previous research, no methods have been created to take previously stated data formats and transform them into graph format where the previous data doesn't explicitly list the nodes (source and target) and/or edges.

Branching this gap between non-network formats with ambiguous nodes to network formats with explicit nodes containing necessary metadata is the first critical step in allowing network analysis to assimilate amongst its tabular peers. Sports data was chosen since there are readily available and easy-to-use Application Programming Interfaces (API) that contain large amounts of sports information (Sportsipy). [1] This information includes statistics on a given sport, team in that sport, or player on that team. The overall contribution of the project will provide insight into the transformation process as well as help promote the movement towards the network conversion of all datasets to graph datasets without the explicit knowledge of source and target nodes.

III. Proposal

A. Initial Information

There are two different scenarios where this project will be deployed: a Command Line Interface (CLI) or an Application Programming Interface (API). A CLI is a service that provides an interface specifically in the command line terminal of the user's operating system. It should be noted that all interfaces from this project will all be in the programming language Python. One example of a Python CLI helper is Typer [2]. Typer makes creating command line interfaces with ease by providing already written functions to enact a fully customizable CLI and generating help page documentation where users would have initially had to define both of these points on their own without Typer. This type of package can be thought of as Platform as a Service (PaaS) [3] since the user is interacting with a program locally, but receiving data that is not managed by the user.

For an API, one example for production and deployment in the Python programming language is FastAPI [4], the API relative of Typer. FastAPI

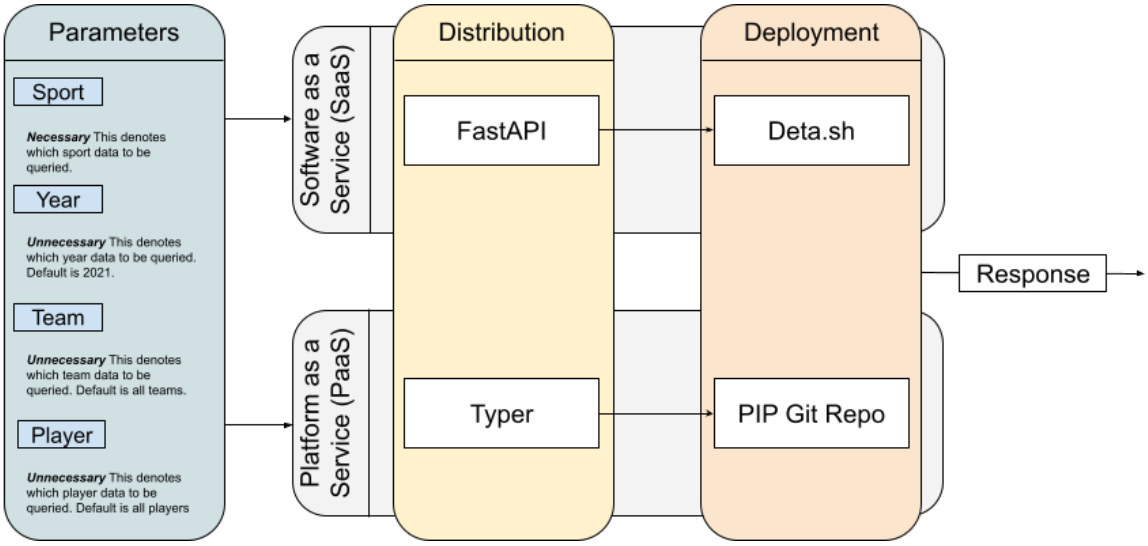

Fig. 1: An overall plan for how the service will be provided. The user will input general parameters, then based on which service will be better, the following service protocols will be enacted.

provides functions to enable users to accurately and quickly develop APIs without the hassle of purely defining every function necessary. Both of these options provide flexibility for the user to fully customize the dataset that is desired. This example can be thought of as Software as a Service (SaaS) [5] since the user does not interact with any any part of the code, just the URL to extract the data.

From this, we can see that the differences in services are trade offs of each other. Therefore, the overall plan will include both of these services enacted as an entire interface but the choice is unnecessary to decide which service is used to start the project.

B. Approach

After deciding the initial service, CLI or API, the process of how the data will be delivered to the user will be crucial. There are certain questions that need to be answered before the plan can be established. These questions will provide support that is needed to distribute the data to the user in a clean and efficient way:

1) Speed: How fast will data be sent to the user?
2) Size: How large is the size of the data being sent to the user?
3) Consistency: If the program is run in the same scenario but different teams, will it roughly contain the same attributes?
4) Scalability: As games progress and more data is added to the underneath API, how will this program react to an increase of data?
5) Temporality: If data is requested for multiple years, in what format will this be delivered?
6) Availability: How many different types of data will be provided from ESPN? (Sports, Teams, Players, etc...)

1) Speed: Fortunately, FastAPI and Typer are written by the same developer so the speed concerns that come with both an API and CLI were initially taken care of. [2] [4] For a CLI, this would require the user to install it on their distribution of Python which means that the speed aspect of this would be reliant on the users internet connectivity. As for the API method, the tool would need to be deployed on a site like Deta. [6] Deta is the recommended client by FastAPI that “hosts” your API for you free of charge by hosting the API in a cloud based system. While this won’t require installing ESPN

on the user's system, there will be minimal latency when connecting FastAPI to Deta. As for the speed on the initial query side from Sportsipy, there will be a wall that will be created between ESPN and Sportsipy given the extraction of data.

This will be the major concern of speed, but processes like `asyncio` and `multithreading` in Python will play a key role in the optimization process of extracting data by query.

2) Size: When querying teams of certain sports, this data will not be cumbersome for any computer, even across multiple years, since most professional leagues do not have more than 35 teams. But when considering College Sports or the players across any league, this data grows exponentially since the situations assume a many to many relationship and a many to one relationship respectively. For example, college football has 10 conferences with 130 teams total and professional football has 2 conferences with 32 teams total, where both college and professional football teams hold 85 and 53 players respectively.

One way to avoid this, is to only return the players that played in games instead of return the entire roster. This will reduce the amount of nodes and edges, as well as the complexity of the entire graph. As more data is needed given the user's specification on time, the nodes will plateau where as the edges will continue to increase until a plateau is hit as well.

3) Consistency: One very important characteristic of the data will be the consistency of the data regardless of the input variables. For example, say a user wants to look at the 2021 college football season. This format should be very similar to the network of the 2021 college football season except the relationships between nodes will be different. The metadata within the nodes will need to contain consistent data across all scenarios of the ESPN query. But, given the structure of the Sportsipy API response, these concerns are unnecessary since the data is consistent regardless of parameters.

4) Scalability: Scalable data is a necessity in any scenario regarding data quality. But, scaling data within networks can be tricky, especially when the data follows temporal trends, due to how an erroneous node addition can remove the networks integrity and validity. Hence, scaling the data in a temporal manner will be key in allowing the user to experience a fully customizable integration. Scaling in a temporal manner will consist of either adding data by a weekly or yearly basis given the parameters by the user.

5) Temporality: Temporality is when any part of the data relates to time. In the case of ESPN, every part of the data extracted will relate to time regardless of the attributes. This poses a problem when multiple weeks or years would like to be extracted by the user. One way to remove this challenge is to provide multiple edges to nodes that already have edges and present the year and week within the metadata. Further consideration is needed to evaluate the best possible process in solving Temporality.

6) Availability: The Sportsipy API only provides the following sports to query from: professional baseball (MLB), professional basketball (NBA) where as of now it is currently not working, professional hockey (NHL), professional football (NFL), college basketball (NCAAB), and college football (NCAAF). Unfortunately, a limitation cannot be fixed until the Sportsipy API includes this data.

C. Plan

An overall plan in Figure 1 is given by how the data will be input to the service and deployed for the network response data. First, the user will input the minimal amount of parameters needed for the API response. Here, the user must state what sport they will like to analyze. If sport is stated to be NFL and no other parameters are given, then the default query is

```
https://extractsportsparts.network/?
sport=NFL&year=2021&team=all&
player=all
```

if the chosen service is an API. If the service was a CLI, then the data would be received from the Sportsipy API and transformed in-house to be saved locally. If the user wants multiple years, then the link will be similar except the year parameter will

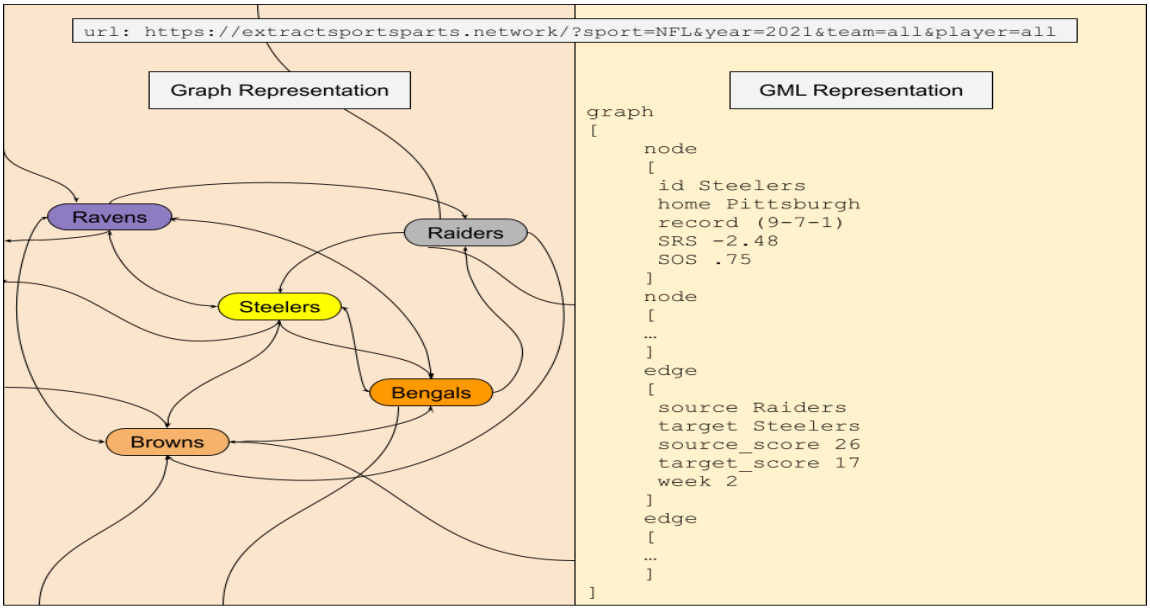

Fig. 2: A specific response given the query url `https://extractsportsparts.network/?sport=NFL&year=2021&team=all&player=all`. Here, double arrowed lines indicate that teams played each other twice.

be `year=2020-2021` instead. These parameters will then be sent to the selected service of choice and transformed into the customized response that the user requested. In Figure 2, a subset of 5 teams from the NFL in the year 2021 is shown. Most have double arrowed connections, which means that they both played at each others home field. The arrow points in the direction that the source team went to go play at the target team's stadium. For example, there is a double arrowed connection between the Ravens and Steelers because they each played each other's home stadium and the week that this game was held is inscribed in the edge attributes. Each edge and node contains metadata about the event and team mentioned respectively. These characteristics are subject to change but these statistics shown in the GML representation of Figure 2 are representative of the type of attributes that will be embedded in nodes and edges.

1) Feasibility: Accumulating the data, scaling it to customized needs, and reporting a consistent format across all permutations of query parameters are necessary for the service to operate in a fully functioning manner. While there may be a few caveats like how professional baseball teams play multiple teams multiple times over the course of the season which would create multiple directed edges in the same direction, this still will not pose a threat to data consistency and feasibility.

IV. Conclusion

This project will create a new path for transforming data that do not explicitly state the source and target nodes and output a network data format for exploration. Even though this service will provide sports data as network, there is no limitation as to what can provided with this framework given the format the API/CLI currently has. Ideally, this program would have a built-in API to query the data rather than relying on an external API to decrease latency between queries. Allowing for growth in this field will allow for novel analyses, and exploration for new data formats and interfaces for extracting data in a network format.

REFERENCES

[1] R. Clark, "Sportsipy: A free sports api written for python." [Online]. Available: https://sportsreference.readthedocs.io/en/stable/

[2] S. Ramírez, "Typer." [Online]. Available: https://typer.tiangolo.com/

[3] W. Y. Chang, H. Abu-Amara, and J. F. Sanford, *Transforming enterprise cloud services*. Springer Science & Business Media, 2010.

[4] S. Ramírez, "Fastapi." [Online]. Available: https://fastapi.tiangolo.com/

[5] E. Ferrari, "Access control in data management systems," *Synthesis lectures on data management*, vol. 2, no. 1, pp. 1–117, 2010.

[6] A. Computing UG, "Deta.sh." [Online]. Available: https://docs.deta.sh/docs/home