UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Project Architecture

Semantic Data Management

Project Group:
Andrea Armani
Ricardo Holthausen
Jesús Huete
Dimitrios Tsesmelis

Professor(s):
Oscar Romero
Petar Jovanovic
Besim Bilalli

Project Period:
Winter Semester 2020

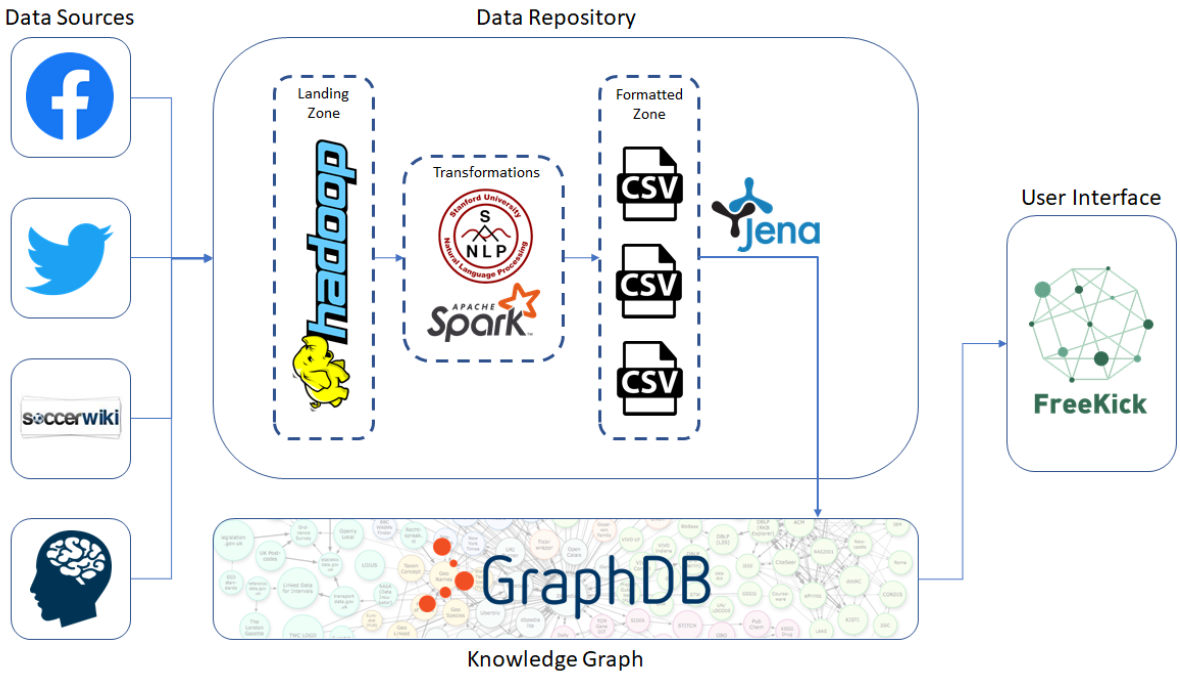
Date of Submission:
June 18th, 2020

Edifici B6 del Campus Nord, C/Jordi Girona, 1-3, 08034 Barcelona

Table of Contents

Purpose Statement

In our project we decided that the best alternative to be used to model and store our data is a Knowledge Graph. To justify such a graph-based solution, we first need to define the functional architecture of our product, which is depicted in the following diagram.



In the above diagram it is depicted the functional architecture of our product. On the left hand side, there are the data sources of our system. In most of the cases, the inputs are in JSON format. This data is directly stored inside our Landing Zone without being processed. They are part of the Batch layer of our application as the collection of such data is done periodically. For instance, let's take as an example the collection of the players' demographics and statistics which is coming from scraping. In this case, this operation should be performed at least once per year as the information of the player may change from one year to another, taking into account that the duration of the contract that a player has with a team varies from one to a couple of years. Moreover, this operation is done in batch mode as we will massively update the information for all the players rather than entry. By storing the data in raw files, it will allow us to integrate it later with new sources of data without needing to modify any schema, as well as generating new on-demand views as requested by the users.

The data that we collect is coming from heterogeneous sources and hence a pre-processing step is needed. That is why, right after the storage of the input files from the source system, we need to apply some transformations and prepare the data in such a way that it will be ready to be imported to the Knowledge Graph later. The main transformations that we apply in this step are done with Spark and by applying Natural Language Processing (NLP) techniques from Stanford University to generate

Sentiment Analysis [1]. In the next step, the processed data are stored in the Formatted Zone in csv files. Within the Metadata Management System we will link the different pre-processed files to generate a global schema of our data. The tool will utilize these mappings to generate the on-demand views which will be consumed by the user interface within the Transformed Zone.

The main reason that made us decide to use Knowledge Graph to integrate our data in a single storage engine was the existence of different data sources and the heterogeneity of the data. Specifically, the fact that the main concept of the data source is the **football player** implies that at some point in our architecture, we have to integrate the data in a single point, where the same player from the different sources will be considered as the same one. In addition, Knowledge Graphs improve the quality of our data, as we are able to link them with other ontologies and add semantics that can be later exploited to produce more meaningful results. For instance, as we have already described, we are using the players' data from social media and dbpedia includes the concept of Facebook and Twitter. Hence, we can link our data with these classes and directly use their properties without the need of redefining them.

Graph Family Justification

To proceed with the project, we were given the task of selecting which type of graph is going to be used to store our processed data and utilize for our application. We have two options: Property Graphs or Knowledge Graphs. Each one has features that would benefit different use cases, and for us the followings are the key features for both:

Property Graphs	Knowledge Graphs
It allows us to uniquely identify instances of relationships of the same type. (Messi) -[plays_at] -> (Barcelona), can be instantiated several times to denote different times at which Messi played at Barcelona.	Each relationship is only represented once. Though you can create several ABOXs with the same relationship, the query would return a single triple <Messi plays_at Barcelona>.
You can qualify instances of relationships. The relationship "played_at" could have attributes such as "season", "year", "manager", etc.	You cannot assign attributes to a relationship. For handling such scenarios, one would need to apply workarounds (reification, singleton property, or modeling workaround).
Apply graph algorithms already built on top of the graphs. Such algorithms allow to extract more information from the graph than the one stored already.	Such algorithms are not by default built for Knowledge Graphs and cannot be used to its fullest.
There are no global unique identifiers. All instances are only locally identified within your graph. Big limitation when looking to integrate with other systems.	URIs are globally used between all Knowledge Graphs and are uniquely identifiers. Makes easier the workload for integration with other systems.
Federation is only provided over that local graph	You can apply a query federation to run multiple

and runs in memory.	queries over different databases. Write a SPARQL query, which with other standards (W3C or R2RML) can translate the query for the corresponding DB.
There's no semantics applied over the data stored. The semantics need to be handled from the application side.	You can apply semantics over the data, which allows users to understand and retrieve the required data in a more intuitive and friendly way.
There's no inference mechanism. What's defined within the graph structure is what exists (close-world assumption).	Inference can be applied over queries to retrieve information that would otherwise be impossible unless it's explicitly defined within the graph.

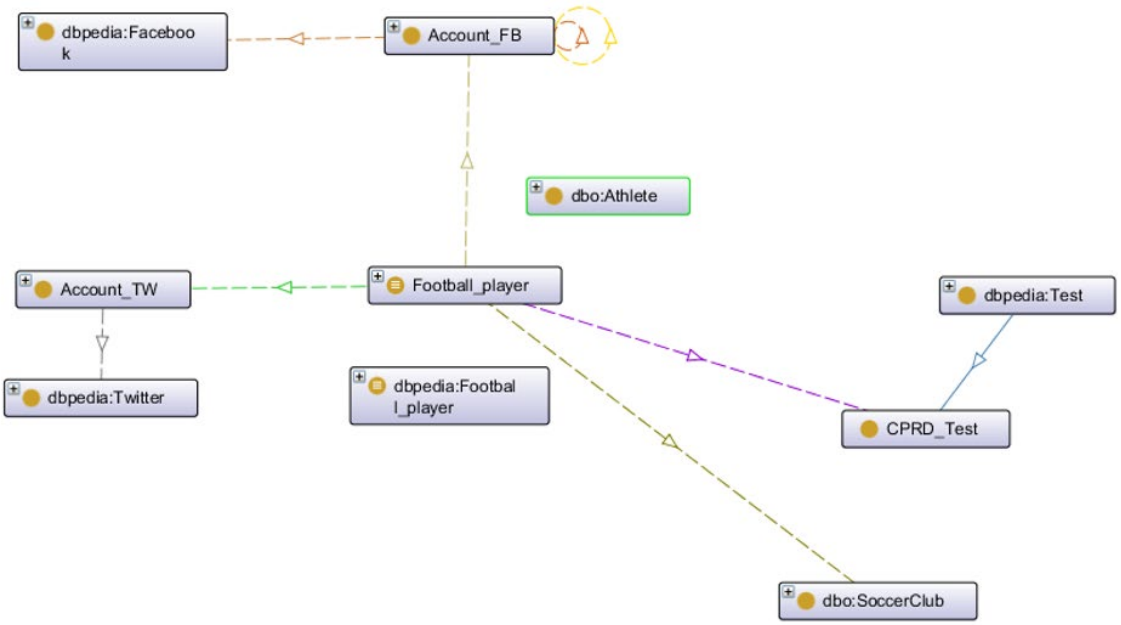
After doing such comparison, we decided to move forward with **Knowledge Graphs**. Specifically talking about the data we're recollecting from the different sources, we don't have an actual scenario where graph algorithms could provide information that would be useful for the application. Furthermore, there's no need to uniquely identify instances of relation types, or apply attributes over such relations. This means that the main benefits of the Property Graphs would not bring any added value.
On the other hand, we're integrating data from Facebook and Twitter. Such well-known social media platforms have ontologies created already within dbpedia which we can reuse for our own graph. Also, by applying some semantics on the data, our queries will become simpler for the UI of the application.
Overall, because the purpose of this graph is more related to data integration, rather than exploitation of the graph algorithms, we believe that Knowledge Graphs are a better option to proceed.

Graph Design

To construct the TBOX, we use the Protégé ontology editor (available for download at: https://protege.stanford.edu/) which is a free, open-source software written in Java, that facilitates the creation of TBOX triples though a user-friendly interface.

All definitions explained below, can be found in full statements in the file **src/knowledge_graph/rdfs_scripts/football.owl** exported from Protégé and attached with this report.

The visual representation of our TBOX created using OntoGraph of Protégé, showing the subclassing relationship among ontology concepts:



In a few words, the main concept in our ontology is **Football_player** which is equivalent to class with **dbpedia:Football_player**. Apart from the data properties that a player has (not shown in this screenshot), a player can be connected to other classes related to social networks, namely **Account_FB** (his Facebook account) and **Account_TW** (his Twitter account). The first two classes represent the accounts that a player owns in the several social media and they are used to hold the data related to their social profiling information (though data properties). Account_TW and Account_FB are connected to **dbpedia:Facebook** and **dbpedia:Twitter** classes, respectively, through the object properties social_network_fb and social_network_tw. In the case of Facebook, a player is considered to have favorite athletes and favorite teams and hence, two additional object properties have been created, the domain and range of which is Account_FB.

Moreover, a player takes several **CPRD_Test** which is a subclass of **dbpedia:Test** class and a player also plays_in (object property) a football team, represented with the dbpedia:SoccerClub class.

Flows for Graph Population

Regarding the flows of the data, we will depict the process followed to collect data related to the football players that we include in our product.

SoccerWiki Scraping

In our product we are interested in differentiating the players according to the league and the country that they are currently playing. That is why, the main program of our scraper receives as an input the link of the league that we want to scrap (e.g. Spanish

leagues). The next step to be done is to list all the teams that belong to this league and call another parser that scrap all the players of each team. The parser of the player is the part of the spider that generates the data and outputs them to a JSON file. The collected information and the transformations that are done in this step are the following:

1. Get the player's demographics ("Full Name", "Club", "Age", "Nation" etc...)
2. Get the current player's rating
3. Get the attributes of the player and flatten (transformed from list of attributes to different columns)
4. Get the list of positions and extract only the first one
5. Get his prefered foot

The next step that we need to do is to clean and transform the produced data. Such transformations are done with Apache Spark. The output of this step is written in CSV files.

Finally, the produced data need to be transformed in a way that they can be imported to our Knowledge Graph. This job is done with Apache Jena, which is basically Java code that receives as input the previously created CSV file and outputs triples that represent the Abox. The triples that we generated at this step are related to the Data properties of the football players. For instance, below there is a sample triple of the output:

```
<http://www.semanticweb.org/ontology/football/Oleg_Reabciuk>
<http://www.semanticweb.org/ontology/football/league> "Primeira Liga" .
```

This tiple means that **Oleg Reabciuk** is playing in the **Primeira Liga** league.

Twitter API

In this specific case, we're interested in retrieving the tweets and mentions from the different football players being scouted. Thus, we don't require to do it in a streaming approach, as it doesn't make sense to wait for the players to send tweets. We will therefore retrieve the player's timeline and mentions utilizing the *getUserTimeline*. This will retrieve a JSON document with an array of the tweets within the timeline.

Using Apache Spark, we created a JAVA project which retrieves the text from the tweets and mentions from the player and applies the NLP Sentiment Analysis algorithm developed by Stanford University [1]. This algorithm will define a number grading for the text from 0 to 4 (Very Negative, Negative, Neutral, Positive or Very Positive respectively).

After doing such transformation, the data needs to be ingested into the Knowledge Graph. For such a process, we generate the triples associating the **Football_Player** to its corresponding **Account_TW**. Then we generate the triples for such sentiment scores as a literal for the property **sentimentScore**. Finally, we must create the triples that give semantics to our accounts, by associating the accounts to the **dbpedia Twitter** class. Examples of such generated ABOXs are displayed below:

```
<http://www.seman      ticweb.org/ontology/football/DylanAsonganyi>
<http://www.semanticweb.org/ontology/football/sentimentScore>
"2"^^<http://www.w3.org/2001/XMLSchema#long>

<http://www.semanticweb.org/ontology/football/DylanAsonganyi>
<http://www.semanticweb.org/ontology/foot                ball/social_network_tw>
<http://dbpedia.org/resource/Twitter>

<http://www.semanticweb.org/ontology/football/DylanAsonganyi>
<http://www.w3.org/1999/02/22          - rdf  - syntax  - ns#type>
<http://www.semanticweb.org/ontology/football/Account_TW> .
```

Facebook API

In order to retrieve data from Facebook we would need to deal with Facebook Graph API. However, there are some limitations both on the number of requests that can be done per hour and user. Besides, in order to gather information from a certain user, we need that user to actually give permissions to our app. For these reasons, we decided to generate dummy data in the same format as the Facebook Graph API.
Specifically, for our proof of concept, we are interested in gathering information about the teams and athletes that players admire. We will generate facebook data for a subset of the football players, and for each player we will assign some football teams and athletes they may have *liked* on Facebook. Other possible operations with this same source could have been performing NLP on the users' posts or obtaining psychological data from all the users' likes and cross that information with data from psychological tests. Nonetheless, as we were already going to perform NLP on tweets, and given the fact that we are generating the data, the information obtained regarding personality traits could be misleading, so we decided to focus on integrating the information regarding sport interests from users.
In a similar way to the Twitter source, we are not treating Facebook API data in a stream manner, given the fact that our application is assumed to gather information regularly (e.g.: daily or weekly), rather than continuously.
Once we have the data in the same format as Facebook would provide us with it, we need to organize this information in first normal form. For this purpose, we need to flatten the list of json documents in a csv file. Once done, we needed to ingest this data in the Knowledge Graph. Apache Jena was used for this purpose, and the players were linked to facebook accounts that can have favorite teams and athletes facebook pages, that belong to actual teams and athletes. Some examples of these triples can be seen below.

```
<http://www.semanticweb.org/ontology/football/Lifumpa_Yande_Mwandwe>
<http://www.semanticweb.org/ontology/football/has_account_fb>
```

```
<http://www.semanticweb.org/ontology/football/1230000000048> .

<http://www.semanticweb.org/ontology/footb                all/1230000000048>
<http://www.w3.org/1999/02/22          - rdf  - syntax  - ns#type>
<http://www.semanticweb.org/ontology/football/Account_FB> .

<http://www.semanticweb.org/ontology/football/1230000000048>
<http://www.semanticweb.org/ontology/football/social_network_fb>
<http://dbpedia.org/resource/Facebook> .

<http://www.semanticweb.org/ontology/football/1230000000048>
<http://www.semanticweb.org/ontology/football/has_favorite_team>
<http://www.semanticweb.org/ontology/football/2022049601386560> .

<http://dbpedia.org/r          esource/Borussia_Mönchengladbach>
<http://www.semanticweb.org/ontology/football/has_account_fb>
<http://www.semanticweb.org/ontology/football/2022049601386560> .
```

Psychological Tests

Using Freekick a player has the possibility to take tests that aim to contribute to the development of his/her psychological profile. We interviewed a sport psychologist that explained us that among the several psychological tests that have the purpose of identify the behaviour and personality of an athlete, the CPRD Test (*El cuestionario <<Características Psicológicas Relacionadas con el Rendimiento Deportivo>>*) [2] is the most successful, so for the proof of concept we will simulate the results of such test. However, there are other meaningful tests that could be provided to the athlete, like the Big-5 Personality Test [3].

When a footballer submits a test, a JSON file is created. Such a file contains the name of the athlete, all the questions and their marks. CPRD measures 5 different behavioral aspects and each question on the test belongs to one of these categories. Therefore there is the necessity to aggregate the questions based on the category they belong to. Such operation is done using Java and the Spark library. The output of the preprocessing step is a CSV file that has, for every player, the results of his/her score in every category.
Once the CSV file is created, it needs to be ingested into the Knowledge Graph. For such a process, we use Java and Jena library. Each test becomes an instance of **CPRD_Test**, a concept in our ontology that is a subclass of **Test** in the Dbpedia ontology. Then we link a test with the player that filled it by linking them through **Takes**, an object relationship that belongs to the ontology we defined. Finally, each category of the CPRD test is represented by a property in our ontology.

```
<http://www.semanticweb.org/ontology/football/cprd_test_Lifumpa_Yande_Mwand
```

```
we> <http://www.semanticweb.org/ontology/football/Mental_ability                    >
"2.7844443" .

<http://www.semanticweb.org/ontology/football/cprd_test_Lifumpa_Yande_Mwand
we> <http://www.w3.org/1999/02/22          - rdf  - syntax  - ns#type>
<http://www.semanticweb.org/ontology/football/CPRD_Test> .

<http://www.semanticweb.org/ontology/football/Lifu                  mpa_Yande_Mwandwe>
<http://www.semanticweb.org/ontology/football/takes>
<http://www.semanticweb.org/ontology/football/cprd_test_Lifumpa_Yande_Mwand
we> .
```

All the flow diagrams are available on Appendix 2

Graph Exploiting Processes

Freekick aims to provide a different scouting process to small and medium teams. In order to do so, it does not only focus on the physical aspect of an athlete, but it also considers a number of other sources to provide a more comprehensive analysis.
Since teams have different needs in terms of player to scout, we have to create different queries to satisfy such demands. Following we list a number of query examples that can be answered using Freekick's knowledge graph:

1. "HIDDEN GEM": a player that is ready to join the team and that is currently playing in a lower division. Although his/her physical ability is not very high, he/she has the right behavioral traits to be a good player at higher level;
2. "EARLY BIRD": young athletes (from 13 up to 16 years old) that, despite not having good physical performance, they have the mental asset to become a good player. The idea is that it's relatively easy to shape the body, but it's hard to change the mindset;
3. "MENTOR": a player that is very experienced so that he/she can help others to develop to their full potential. The features that are desirable in such scenario are a high team cohesion, stress control and a positive sentiment score;
4. "BAD BET": Player that despite having a high physical ability, is not motivated, has a bad mental ability, team cohesion or negative sentiment score.

Among the previous queries, the first two are designed to discover valuable assets to add to the roster. Such queries aim to be an optimization of the usual scouting process. As for the third and fourth query, they are something that currently do not exist.
Identifying the right MENTOR to support the development of a young group of players is very difficult for low budget teams: there are no available metrics to assess how an experienced player would fit in a young team.

Finally, when a small team has to build the roster, it faces a relevant problem, which is the limited budget to acquire players. For instance, a club could decide to invest a big part of its budget to acquire a player with astonishing physical skills. However, if the athlete has behavioral issues and does not get along with the rest of the team, the investments would not only be useless, but it would also affect the performances of other footballers. Medium to small clubs are heavily affected by the decisions they take during the market sessions and nowadays they do not have a way to support them, beside comparing players statistics.

Metadata Generation, Storing, and Re-use

Proof of Concept

As stated throughout this document, our proof of concept consisted on integrating information from several, different sources into a Knowledge Graph and being able to perform queries that could be of interest for a football scouting department.
When dealing with football data, the information can be obtained from different places. We chose four different data sources (Facebook, Twitter, Soccerwiki and questionnaire results), as they provided us with information from different fields that can be of interest when researching for football players' capabilities. This data is available in semi-structured format (JSON) and had to be processed in different ways (v.g.: Tweets had to be analyzed using NLP to obtain their sentiment, Soccerwiki data had to be cleaned, etc.).
Once the data was preprocessed and available in CSV format, it had to be ingested to our Knowledge Graph. To do so, the Java library Apache Jena was utilized, and each of the CSV files generated in the formatted zone were turned into sets of triples suitable for being imported in GraphDB.
Before ingesting our data to the Knowledge Graph, it was necessary to build our TBOX. Protégé was used for this purpose, and the ontology created contained the different relations between the data from the four original sources. Finally, the TBOX and the ABOX were imported to GraphDB. Then, making use of this database interface, several SPARQL queries were tested, thus achieving our original goal.

In this link there is the complete dataset that we used for our PoC.

References

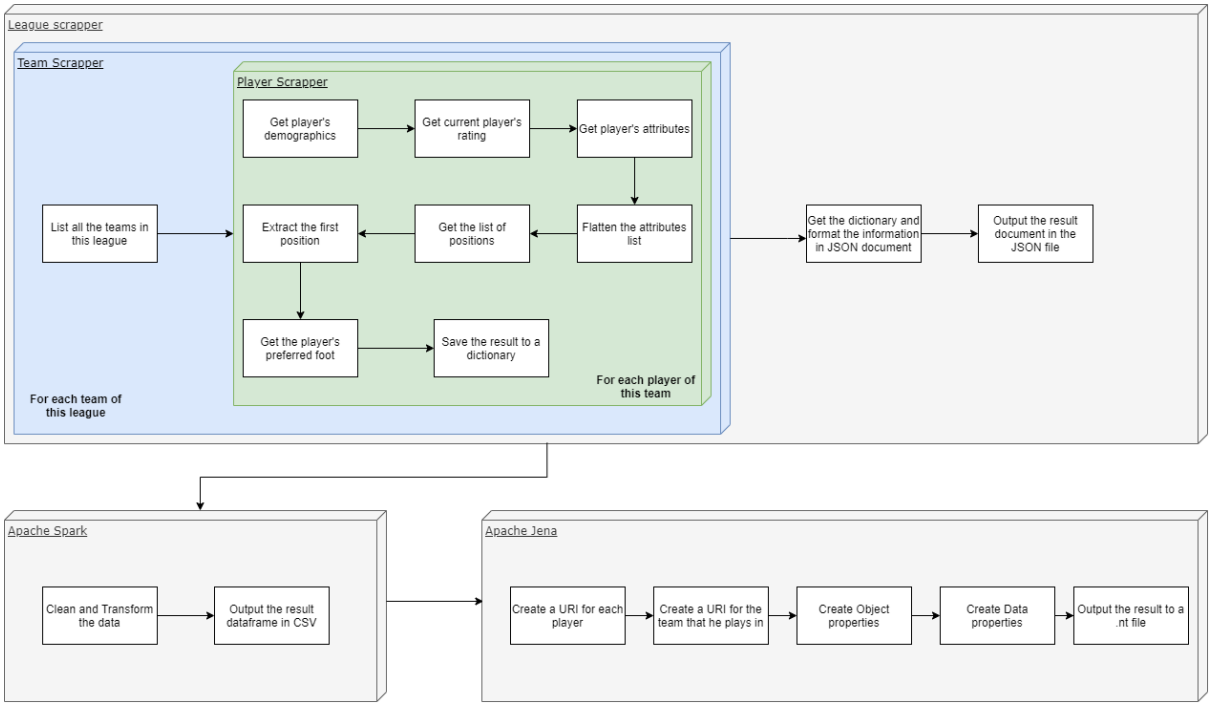
[1] Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60. [pdf] [bib]

[2] http://www.scielo.mec.pt/pdf/aps/v19n1/v19n1a09.pdf

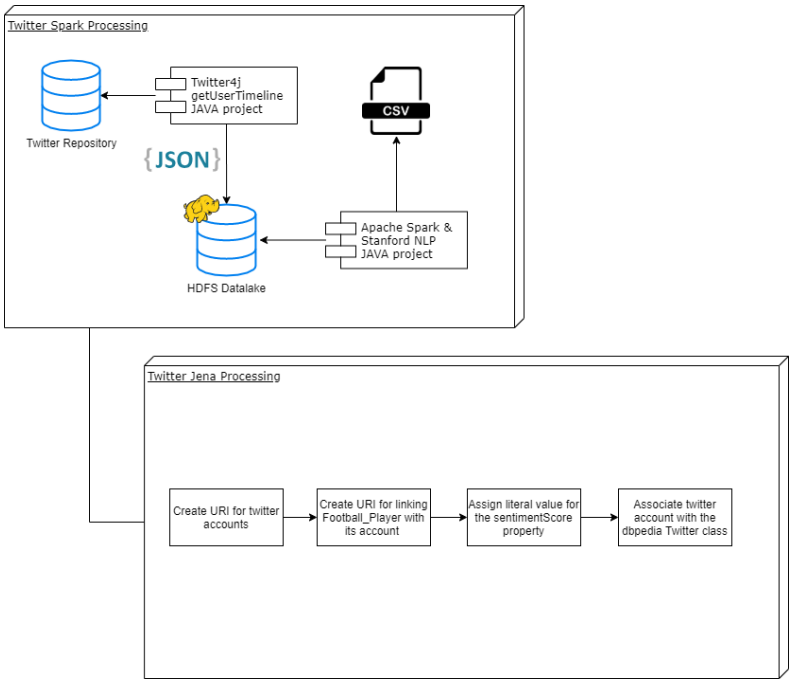
[3] Goldberg, Lewis R. "The development of markers for the Big-Five factor structure." Psychological assessment 4.1 (1992): 26

Appendix 2

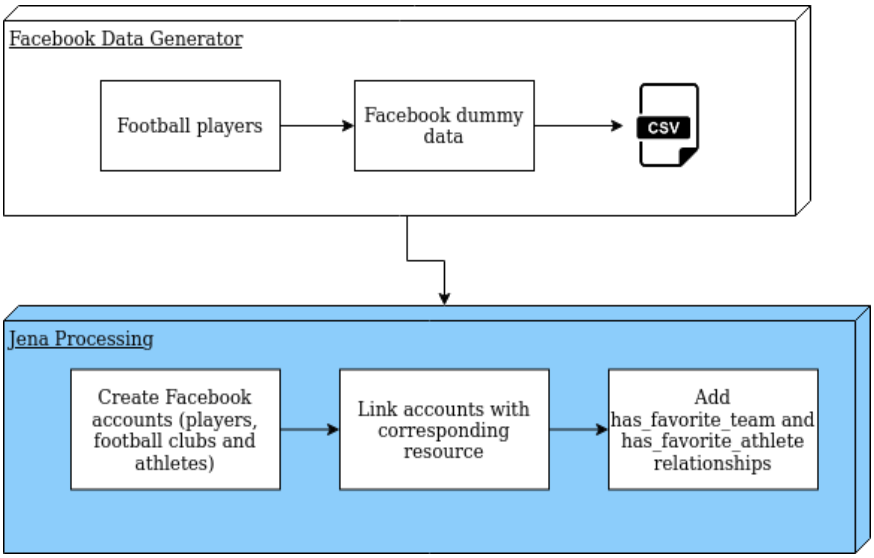
Soccer players data



Twitter data generation



Facebook Data Generation



CPRD Test Data Generation