English Premier League Information Platform

Diogo Torres
FEUP
Porto, Portugal
up201506428@fe.up.pt

João Damas
FEUP
Porto, Portugal
up201504088@fe.up.pt

ABSTRACT

This work deals with the collection, refinement, analysis and querying of datasets regarding the English Premier League in the last five years. Their goal is to create a dataset that allows answering non-trivial queries that are not feasible in traditional sources. For this, different data sources are investigated in order to extract information about all matches across seasons, including lineups and match events. The data is then put through a refinement process, conceptually described and ultimately analyzed using exploratory techniques. Early results show patterns such as a rise of events at the end of each half and a growth of match report size. Through the usage of Solr, an information retrieval tool, we define a collection and respective documents and describe their indexing process. Some of the previously identified retrieval tasks are then answered, while also exploring some of the tool's features, and evaluated according to their precision. Results obtained are satisfactory for the expected system user profile and are consistent with existing literature. Finally, using Semantic Web tools, we define and populate an ontology that reuses existing concepts and answer the remaining retrieval tasks identified, while exploring the SPARQL query language. We also show how to integrate with external collections in other ontologies. Results were as expected and allow to see Semantic Web and Information Retrieval as distinct areas, each with their own use case. However, Semantic Web tools often present limitations due to their lack of a standard implementation or beginner friendliness.

CCS CONCEPTS

• **Information systems** → **Data cleaning**; **Search engine indexing**; **Relevance assessment**; **Semantic web description languages**; *Document representation.*

KEYWORDS

datasets, scraping, football, premier league, data refinement, data analysis, information retrieval, solr, indexing, information system evaluation, semantic web, rdf, owl, ontologies

1 INTRODUCTION

The English Premier League is one of the most followed around the world, and its popularity seems ever increasing. It usually never fails to deliver, with recent moments like the Leicester fairy tale, the consolidation of the "Big 6" [16], which always provide quality matches, or the more recent Manchester City/Liverpool dominance. However, in an effort to try to comprehend these phenomenons, complex queries sometimes need to be answered, and it is not always easy to find a platform that contains such properties, especially across several seasons. Hence, the goal of this paper is to describe a system that can help interested people obtain more specific information about historic Premier League data in an easier and faster way.

The remaining of the paper is split into three major parts. The first, regarding data preparation, starts by describing all the data sources and the datasets' merging. Their quality is then assessed and necessary refinements are performed to obtain a final version of the dataset, which is then presented in a conceptual model. The model highlights all the main entities present in the domain, as well as associations between them. After that, the data is put through exploratory analysis techniques in order to have a better understanding of its values, respective distributions and (un)common patterns. Finally, possible retrieval tasks are identified in order to provide a glimpse of the possibilities that the final product is expected to have. The second part deals with the implementation of an information retrieval system. We start by detailing the two versions of the system used for comparison and evaluation metrics, followed by a justification on the tool choice. A description of the collections and their documents follows, as well as the full indexing process. A subset of previously identified retrieval tasks is then answered and the results in both systems compared according to the measures defined. Finally, a short evaluation of the tool reflects the process' easiness and the quality of the results. The third and last part takes a new perspective on the dataset's domain and implements an ontology based on it. We initially reflect on the origins of the Semantic Web, then revisit our conceptual model and analyze what has already been done in our domain. Following this, we describe the creation (i.e., classes and properties) and population process of our new ontology. We then answer the remaining retrieval tasks identified by querying the populated graph, before showing small examples of how the ontology can be integrated with external existing collections. Finally, there is an evaluation of the new technologies explored and a comparison between the different paradigms that were dealt with (Semantic Web and Information Retrieval), before some considerations on the developed work conclude the paper.

2 DATASET PREPARATION

Before finding appropriate datasets, it was necessary to choose a feasible scope and outline what information we were going to store and present. The English Premier League has been around for over 25 years, however our interest in it and the memories associated are much more recent, therefore we concluded that this project would cover the last 5 full seasons as of the time of writing, which already provides the amount of data wanted to work on. On the other hand, since matches are the league's core, they will be the center of the project.

Finding sources on basic match information (i.e., score, teams involved, date) is not difficult, as many sources collect these properties, often what people are looking for. Despite this, it is hard to

find a source that provides datasets ready to be worked on and with complete information for several seasons. After some research, we used data from football-data.co.uk [19], an online betting central, as well as a free match record provider, part of a network of betting platforms for different sports. It is one of the most widely used in the United Kingdom and is updated daily, with the purpose of helping "football (...) enthusiasts analyse many years of data quickly", so the source is seen as current and authoritative. Its advantage over other platforms is that it provides ready to download CSVs (one per season) with information on all matches' date, teams involved, scores, referee and other statistics and betting odds. For this project, not all of the fields are necessary, so some refinement was necessary, which is described in more detail in the next subsection. Hence, we have the first dataset, for basic match information.

Finding reliable sources for more specific match facts, such as lineups, substitutes, match events (e.g., goal scorers, cards) and match reports was not as direct. After some early API experimenting, there were no free options that suited our needs. Therefore, we had to resort to scraping web pages. Early on, an online collaborative platform, Football Lineups [23], appeared as an option. It is very complete and, with a vast online community contributing, it is bound to be reliable. However, its web pages are not very well structured and were deemed hard to scrape for an automated process. Furthermore, we needed to enrich the dataset with information for full-text search. For that, we opted for match reports. In order to obtain them, a British newspaper, The Guardian [20], was initially considered, for having every report necessary in a relatively accessible way. In the end though, another newspaper, SkySports [37], was chosen, providing all the information of the previous two, through generally well structured HTML documents. Being one of the top media in the country, the source is certainly authoritative and, thus, the scraping process took place there. For it, BeautifulSoup [14], a popular scraping library, was mostly used. For reports in particular, a newspaper scraping lib, Newspaper3k [26], was used, since reports don't have a consistent enough HTML structure to be automatically extracted in a feasible way.

From Sky, we then obtained the second dataset, where information on all match squads was extracted, as well as associated events to each player (i.e., (own) goals, penalties missed, yellow or red cards and substitutions). To further complement the basic match information dataset, the arena and attendance were also parsed. As we extracted it directly, only the relevant fields were saved and there were no incompleteness problems. For processing easiness and due to its natural hierarchical structure (game - team - player - event), the data was stored in JSON files.

Both datasets possess information on participating teams and the date for each match, its unique identifiers, and it is through these fields that the datasets are merged.

2.1 Data cleaning and refinement

Although the collected data was clean and complete, it had some inconsistencies across datasets which proved and obstacle to their immediate merging. To fix this, the basic match information dataset had to be refined in order to have a more easily workable output. With this in mind, the OpenRefine [28] tool was used. As it works best with CSV files, it was a natural choice.

The first step was cleaning the data. For that, unnecessary fields were removed. CSV files from football-data.co.uk append a series of columns with general statistics and betting odds for various providers that do not have relevance for this project, hence they were removed from the files. Some columns that remained were renamed for clarity purposes, namely score columns (e.g., FTHG was renamed to HomeFTScore, and similar for half time scores and away teams).

Some refinement was also in order. Sports teams are known by various names and aliases, and not even in their most common name there is usually an agreement when writing. Because of this, most clubs had different namings between the two datasets. While some differences seemed less inconvenient (e.g., Brighton and Hove Albion was named Brighton & Hove Albion in the CSVs), others were completely different, in a short name to full name analogy (e.g., Wolverhampton Wanderers, in the CSVs, is simply Wolves). Luckily, OpenRefine possesses a feature called Reconciliation [32] that matches text with database IDs using Wikidata [48] as a source, which allowed the tuning of the club's names to their most common and complete namings, as stored in the JSON dataset. The process had some manual intervention in the end to clean up the aliases (e.g., remove F.C. from all names), but overall was faster than a manual replace on all instances. As a last step in refinement, dates were in inconsistent formats and were all converted to a universal DD/MM/YYYY.

Table 1 presents a summary of the final datasets. Appendix A provides an example of each one.

Table 1: Summary of collected and processed datasets.

Dataset	Format	Contents
Basic match information	CSV	Match date, teams, half and full time scores, referee.
Match details	JSON	Match date, teams, full time scores, report, arena, attendance, squads, events by player.

2.2 Conceptual data model

The conceptual data model describing the collected data can be found in Figure 1. The Game class assumes a central position, being the main focus of the dataset. A season is identified by its ending year (e.g., 2018/2019 is represented by 19) and has several games. A Game is played in an Arena, is officiated by a Referee and has a host and visiting team, of which it stores half and full time scores. It also contains a Report associated that describes, in natural language, the matche's history. Several events occur throughout the game's natural flow, in any given minute, and are associated with a responsible Player. Each player can figure in multiple games for one or more teams, though only one team per game, and a team has multiple players on the field, hence the ternary association, which also includes a boolean attribute symbolizing the player's starting status in the game, being true when he is in the lineup.

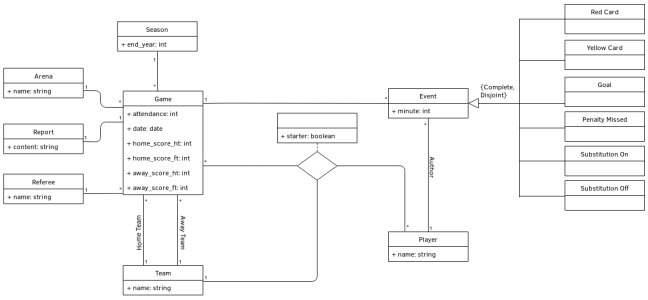

Figure 1: Dataset conceptual model.

2.3 Data characterization and analysis

The project covers the last five seasons of Premier League matches. Therefore, in total there are 1,900 entries, each corresponding to a match played. Each averaged 2.7 goals scored and 3.5 cards given out. Regarding other entities, there are 29 teams present and about 1400 players. Considering three teams are relegated/promoted each season and there are 20 teams per season, 29 teams across five seasons shows its competitiveness. Well over a thousand players helps corroborate the collection's good size to query later.

Some exploratory analysis was also made in order to understand data patterns and assess if they are semantically expected. For this, first, a multi line plot showing the distribution of events throughout the 90 minutes of a match was produced. Figure 2 summarizes the findings. Note that goal count includes own goals and card count aggregates yellows and reds, since one of the two in the pairs always appears in much larger quantities (i.e., own goals represent only 3.1% of all goals and reds 3.8% of all cards). Substitution count only considers entering players, as a player going off is symmetric and, therefore, redundant.

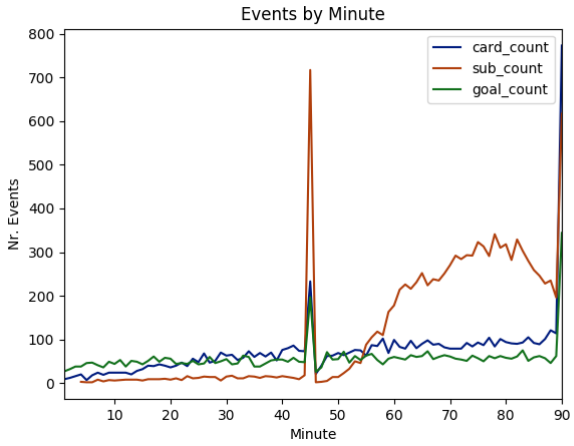

Figure 2: Match event distribution by game minute.

Some common patterns can be observed. First, the big spikes in the 45th and 90th minutes, which are to be expected for two reasons: first, because, being in the end of each half, they encourage teams to exhaust their efforts to keep the result to their advantage (in the case of substitutions, they freshen up the game at half time and help kill time in the end); second, due to, in the dataset, events occurred in injury time having the compensation scratched (e.g., an event occurring in the minute 45+3' is stored as being in the 45th).

On the other hand, the number of events gradually increases during the second half. In a typical match, teams gets progressively tired, therefore committing more fouls and mistakes that lead to goals. Card counts are always generally higher, which is in accordance with the higher average, and substitutions assume much larger values: since each team is allowed 3 per game (and often use all), this is also anticipated.

Another analysis performed was on goal distribution, more specifically goals per month and per season, to understand possible patterns regarding periods with a better or worse scoring form. Figure 3 summarizes these findings.

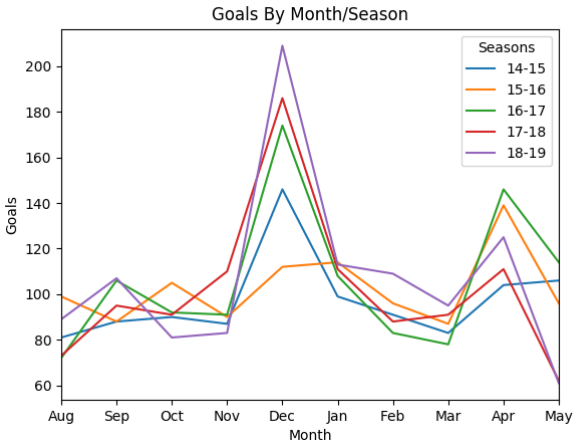

Figure 3: Goal distribution by season and month.

One pattern that sticks out is the higher goal count in the month of December. During this month and, in particular, after Christmas, there is a higher frequency of games, with teams playing every 2 to 3 days instead of weekly. Hence, there is always a bigger number of matches, which culminates in what is observed.

Another month that is also usually prolific is April. During that time, international breaks are done and through. With the final weeks of season approaching, teams start to make a bigger push in pursuit of their season goals.

It should be noted that overall goal counts have been generally increasing, with total values amounting to over a thousand consistently.

Finally, the starting months of a season usually have lower goal counts. For one, teams are still getting into rhythm. Another explanation is the frequent international breaks during these first months, which cause a reduction in the overall number of games in this period (and, therefore, cause a higher concentration e.g., during December).

Another interesting pattern is the number of goals scored by the winning team on each game. The findings for this are summarized

in Figure 4. Note that games that ended in a draw were not included, as there is no winning team.

Figure 4: Number of goals scored by winning team.

The results allow very different conclusions. On one side, a mode value of 2 is reasonable, as both teams scoring is not uncommon and, therefore, one goal is usually not enough. However, a one goal game is present more often than expected, which can be a sign of the defensive sharpness of the teams, something sometimes overlooked in favor of goal scoring form. This is further corroborated by the vast majority of the values lying within the range [1, 3]: teams, however distant on the league standing at the time, can always have close, competitive matches.

Finally, an analysis on match report length, in order to assess coherence and length evolution, was performed. For this, each season was summed up in a box plot, which is shown below in Figure 5.

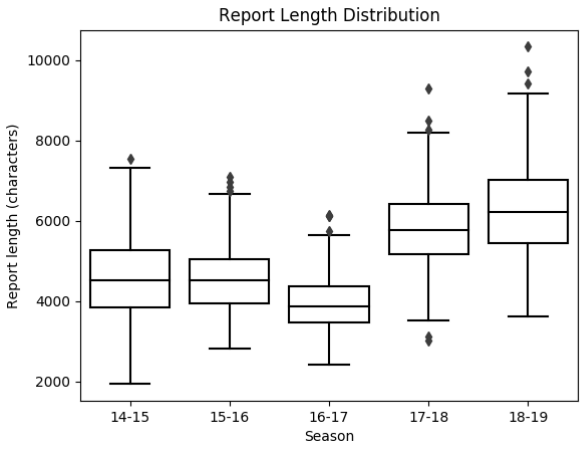

Figure 5: Report length (in characters) distribution.

Results show the evolution in the reports' length, especially in the last couple of seasons. Match reports on Sky haven't always had the same structure and, therefore, can't be expected to have a consistent length throughout time. In more recent years, reports have been more sectioned with further details and insights on each key aspect of a match (e.g., man of the match, managers' say). This justifies the increasing median length observed.

Outliers are not very common and normally correspond to unusually important games. Take the largest value in the 2017/2018 season as an example: it corresponds to the match report from Manchester City 2-3 Manchester United. This was the most important match of the season because 1) it was a Manchester derby 2) Both clubs were in the top 2 league spots at the time and 3) If City won, they would become champions. United turned the score around, which shocked the world and, ultimately, gave birth to extensive reports and analysis.

2.4 Retrieval tasks

With the collected information, different types of information or statistics can be queried. General statistics are collected by various platforms (e.g., top scorers of a season, most carded players, most subbed on player, most appearances). One example is the official Premier League page [43]. Therefore, for a subset of retrieval tasks, there exist similar services. However, when we start combining multiple statistics, or restricting them to more specific time frames, like minutes in one or across several games, it becomes harder to obtain this information in a timely manner.

Some interesting retrieval tasks identified are:

- Which team wins more often with Mike Dean as the referee?
- Which team is most carded during second halves?
- What is the most common substitution in Tottenham Hotspur during 2017/2018?
- Which players score at least in five different months in all seasons?
- What season has the highest percentage of goals scored in the last 15 minutes of the game?
- Which player scored more goals at Old Trafford?
- Which team let a lead slip the most during 2015/2016 (i.e., scored first, but didn't win the game)?
- What players score the most under big crowds (e.g., games with above 50000 attendance)?
- What is Tottenham Hotspur's win ratio when both Harry Kane and Dele Alli start together, since 2015/2016?
- What games featured at least 3 goals from Sergio Agüero?
- What games have Tottenham Hotspur won since the start of 2018?
- What Tottenham Hotspur home games had Heung-Min Son mentioned, despite him not scoring?
- What Liverpool home games had a goal from Mohamed Salah?
- What matches mention Wolverhampton Wanderers, even if they are not directly involved?

3 INFORMATION RETRIEVAL

Information Retrieval is the process of retrieving a set of documents over a collection (also called corpus) that best suit a given information need. Ad hoc search is the most common retrieval task, where a user specifies their information need through a textual query that is matched against all the documents and matching results are ordered by their score (i.e., how relevant the system deemed them). This is the type of search that is explored in this work.

In order to evaluate results, two systems will be used, a baseline that only performs matching on key fields (henceforth System 1), and one with weights tuned (e.g., by boosting more relevant search fields; henceforth System 2). The top 10 results for each query will be analyzed via their Precision at 10 (P@10) and Average Precision (AP) values. Precision (Eq. 1) at n measures the precision level at a usually lower recall level (i.e., n « collection size). Because Precision at n is a measure that is easily influenced by the number of relevant documents, AP will be used to complement it. AP is defined as average of the precision values obtained for the set of top n documents existing each time a new relevant document is retrieved. It does not depend on recall and is considered a more stable measure. Averaging APs over the different information needs gives us the system's Mean Average Precision (MAP).

$$\text{Precision} = \frac{\text{Nr. relevant documents retrieved}}{\text{Nr. documents retrieved}} \tag{1}$$

3.1 Tool selection

The main two tools considered were Apache Solr [38] and Elasticsearch [17]. Both are based in a common core, the Apache Lucene [8]. Solr is a more mature project, therefore its documentation and support are wider. Elasticsearch, on the other hand, is built on a modern stack and works better out of the box, e.g., with JSON formatting, despite Solr also adding support for it since its initial releases. Both tools provide a similar set of functionalities, e.g., full-text search, wildcard searching, boolean queries, boosting, faceting, proximity search, fuzziness, range filters, synonym support, and a well documented REST API for indexing and querying. Perhaps Elasticsearch has the upper hand in scalability, as Solr used to work with a Master-Slave model and only recently introduced the usage of Zookeeper [51] to help scale systems. While scalability is important in an Information Retrieval system, it is not the focus of this work and, hence, the advantage loses its value.

For this, Solr was our choice. Both systems will use the eDisMax query parser [39] supported by the tool. Some caveats still apply: query time multi word synonyms support is recent and not without its trade-offs, and functionalities such as negative boosts have been removed, which calls for workarounds to achieve similar results. These topics are addressed during the retrieval process.

3.2 Collections and documents

A football match defines a document in the system developed. As there will be no other type of documents, there will be a single collection to be queried upon, defined by the 1,900 matches collected.

Previously we described the merging process between both datasets that produce input documents to Information Retrieval tools. However, a structure similar to the one presented in Appendix A is not ideal for usage at this stage. On one hand, Solr's documentation refers only first-level child document indexing and is not clear on an easier alternative to multi-level nesting support. On the other hand, with the current structure, it is not trivial to serve information needs such as retrieve matches where a given player has scored. Therefore, the documents that will be indexed will suffer a change in team squad structure, keeping only relevant information for text based retrieval and storing it plainly: lineup and substitute player names are stored in flat lists. Moreover, events are also stored as flat lists whose elements are player names associated (e.g., a home_scorers field that contains the names of all goal scorers for the home team in the match). Goal scorers are considered the most relevant event of a match and what users would normally think of when searching, hence they will be the only event stored in the documents. With future usage data, one could analyze if the inclusion of other events would represent relevant additions, even if they're not indexed. Finally, dates were converted to ISO-8601 format (YYYY-MM-DD), as required by Solr.

Appendix B provides an example of a document part of the corpus.

3.3 Indexing Process

Not all fields are expected to be searched upon. Thus, the first task was to identify fields that are only present to provide further information on a match when retrieved. These fields include attendance and half and full time scores: while there could be some information needs involving some of these fields (e.g., games with at least x goals), they likely fall within data retrieval instead. Moreover, some of these needs have reasonable textual substitutes (given detailed enough match reports). They are stored using default Solr field type pint with its indexed property set to false. Scores are actually defined using a dynamic field that captures all fields with a name template of '*_score'. This allows an easy removal or addition of additional scores without having to change the schema. The date field is indexed, but stored using Solr's pdate field type, which stores the field using a DocValues [15] structure that is column-oriented and provides faster lookup for aggregation, sorting and faceting, as opposed to an inverted index.

All other fields represent textual data that is subject to an analyzer pipeline that can be configured separately in index and query time. This pipeline includes a tokenizer that takes a raw input and parses tokens by a given rule, and optional filters that apply further processing to each token generated. While Solr provides default field types for textual fields (either general or language specific) with set up analyzers, they did not meet all of our needs: they are either too specific, or too broad. Moreover, we wanted to explore available options and, as such, defined four field types, one for each text type and all based on Solr's TextField: clubNameField, playerNameField, arenaField and reportField.

The first three use a whitespace tokenizer that splits text on whitespace characters. Reports are split using the classic tokenizer that also treats punctuation as delimiters and discards them, but has looser delimiter rules (e.g., does not automatically separate on hyphen). Filter wise, a lower case filter that converts all tokens to lower casing is common to all field types. This facilitates search by preventing non matches due to casing mismatch. In player names,

an ASCII folding filter was also applied. This filter converts alphabetic, numeric, and symbolic Unicode characters to their ASCII equivalents, if one exists (e.g., Agüero is converted to Aguero). When searching for a player, accents are often ignored, hence the utility of this filter. Both forms of each token are preserved, so as to ensure there is a match in both cases. The report field also applies this filter, as it also mentions players. Furthermore, it applies two extra filters to remove English stop words and remove possessives (e.g., token "Ronaldo's" would filter out to "Ronaldo"). A stemming filter that uses the Porter Stemming algorithm [29] was also applied to reduce vocabulary size and potentially increase matching.

Query time analyzers for these field types are similar to the index time analyzers described, the only difference being there is a synonym filter injected as well, to provide synonym support both in club and arena names. This does not come at random: index time synonym expansion requires reindexing the whole collection every time the synonym file is changed. However, only recently did Solr gain support for query time (multi word) synonyms. The existing synonym filter would not produce correct graphs for multi word synonyms, as it stored token position, but not position length, causing multi-word synonyms to improperly mix with surrounding, non-related tokens. This was later nicknamed "sausagization" and has been analyzed in detail [34] [33]. The authors point to some possible caveats, such as matching and scoring implications on some query parsers, but agree overall on the filter placement, which is recommended by the official documentation as well. Listing 1 shows some configured synonym examples.

The schema was defined using Solr's Schema (REST) API [41]. The documents were indexed using Solr's Post tool [40] by providing an input JSON array file. Indexing time was about 2.5 seconds and the index occupied around 9.6MB of disk size. Tables 2 and 3 summarize the schema definition.

Listing 1: Synoynms file

```
# Club synonyms
arsenal, gunners
tottenham hotspur, hotspurs, spurs, yids,
    lillywhites
manchester united, man united, man u, red
    devils
...

# Arena synonyms
White Hart Lane, the lane, tottenham hotspur
    stadium
Old Trafford, theatre of dreams
Anfield, kop
...
```

3.4 Retrieval Process

We will now focus on exploring a small subset of Solr's features while trying to answer some of the proposed retrieval queries (see final tasks from section 2.4). For each task there is the identification of the information need, the query thought to be most adequate, the top 10 results for each system, their relevance judgement and performance metrics as previously described.

Table 2: Custom field types defined. Properties in brackets next to filters are configurable arguments which were set to true or a given value.

Name	Tokenizer	Filters
clubNameField, arenaNameField	WhiteSpace	LowerCase, SynonymGraph [ignoreCase, expand] (query time)
playerNameField	WhiteSpace	LowerCase, ASCIIFolding [preserveOriginal], SynonymGraph [ignoreCase, expand] (query time)
reportField	Classic	Stop [words = 'lang/stopwords_en.txt', ignoreCase], LowerCase, EnglishPossessive, ASCIIFolding, PorterStem, SynonymGraph [ignoreCase, expand] (query time)

Table 3: Document schema fields.

Field	Type	Indexed
date	pdate	Y
arena	arenaNameField	Y
attendance	pint	N
home, away	clubNameField	Y
home/away lineup, home/away subs, home/away scorers	playerNameField (multivalued)	Y
referee	string	Y
*_score	pint	N
report	reportField	Y

Information need: Games that featured at least 3 goals from Sergio Agüero

With this information need, we seek to find all the games that featured 3 or more goals from the striker Agüero. We know that he scored at least 3 goals eleven times overall, ten of which in the last 5 seasons [50] (our scope), so a perfect system would return these 10 games in their top 10.

The query chosen is [*"Sergio Aguero" hat-trick*]. For one, the phrase query guarantees that we match the player Sergio Agüero and not other Sergio, per example. We then chose *hat-trick*, since, to score at least 3 goals, there is a hat-trick involved, and this is usually mentioned in the match report.

Feature wise, System 2 will explore proximity search boosting: documents where "Sergio Aguero" and "hat-trick" appear near each other will have their score boosted as they will probably fit the information need better. For this, the report field was given a factor

10 boost if both appear near each other (with a max token distance, or slop, of 4). Tables 4 and 5 summarize the configuration and results for this query, respectively.

Table 4: '"Sergio Aguero" hat-trick' query parameter configuration.

	System 1	System 2
Parameters	qf=home_scorers away_scorers report	qf=home_scorers^5 away_scorers^5 report pf=report^10 ps=4

Table 5: '"Sergio Aguero" hat-trick' query top 10 results (R = Relevant).

	System 1		System 2	
Rank	**Game**	**R**	**Game**	**R**
1	Man City 3-1 Arsenal 18/19	Y	Man City 3-1 Arsenal 18/19	Y
2	Man City 3-1 Newcastle 17/18	Y	Man City 3-1 Newcastle 17/18	Y
3	Chelsea 0-3 Man City 15/16	Y	Chelsea 0-3 Man City 15/16	Y
4	Man City 4-1 Tottenham 14/15	Y	Man City 6-0 Chelsea 18/19	Y
5	Man City 6-0 Chelsea 18/19	Y	Man City 6-0 QPR 14/15	Y
6	Man City 6-0 QPR 14/15	Y	Man City 4-1 Tottenham 14/15	Y
7	Stoke 1-4 Man City 16/17	N	Watford 0-6 Man City 17/18	Y
8	Man City 6-1 Huddersfield 18/19	Y	Bournemouth 4-0 Huddersfield 17/18	N
9	Man City 5-1 Leicester 17/18	Y	Stoke 1-4 Man City 16/17	N
10	Man City 5-1 Bournemouth 15/16	N	Man City 6-1 Huddersfield 18/19	Y
P@10	0.8		0.8	
AP	0.9705		0.975	

There were a total of 343 games returned. Results show expected differences. Both systems achieved a P@10 of 80%, with non-relevant results at the second half of the list. Although these games do not satisfy this information need, in one Agüero scored two goals, closing in on a hat-trick, which was then mentioned in the reports, and the other featured a hat-trick from another Man City player. In System 2, with field and phrase boosting, results are just slightly better. What may come as a surprise is the 8th result, which does not feature Agüero nor Man City. However, upon manual inspection, we observed that the report mentioned several times the hat-trick of a player in that match. This term frequency is probably what boosted the game to the top 10. Overall, the union of both systems found 9 out of the 10 games, which should satisfy the user's need.

Information need: Games that Tottenham Hotspur won since the start of 2018

With this information need, the goal is to obtain games that ended in a Tottenham victory. Games are expected to be from 2018 or later. Tottenham have won at least 10 games in that period, so a perfect system should return 10 games in the top, all corresponding to wins.

The query chosen is [*spurs AND (win OR victory)*]. *Win* and *victory* are the two most used terms to describe a team overcoming another. To explore both possibilities, we make use of Solr's boolean features. This helps matching both expressions, especially when applying phrase boosting to try and extract more relevant results.

Feature wise, both systems will explore, as described, the usage of boolean operators. Phrase boosting, introduced in the last information need, will also be applied, under the same conditions. When searching the report field, which is natural language text, phrase boosting is suitable, as it is where expressions can be found. There is also the usage of synonyms: Spurs is one of the used nicknames for referring to Tottenham Hotspur. Finally, range filters are also applied to limit results to the desired period (since the start of 2018). Tables 6 and 7 summarize the configuration and results for this query, respectively.

Table 6: 'spurs AND (win OR victory)' query parameter configuration.

	System 1	System 2
Parameters	fq=date:[2018-01-01T00:00:00Z TO *] qf=home away report	fq=date:[2018-01-01T00:00:00Z TO *] qf=home^5 away^5 report pf=report^10 ps=4

There were a total of 67 documents retrieved. The combination of the intersection operator (AND) and the filtering of results by date led to this lower amount than other information needs. Results are very similar across systems. The top 5 documents are equal and there are no unexpected differences in the other half. All non-relevant results retrieved correspond to games that Tottenham were controlling (and sometimes winning) but ultimately let the points slip, hence their relevance assessment by the system is understandable. The top result is somewhat surprising by its non-relevance. Upon manual inspection, the report mentioned Tottenham at least a dozen times, perhaps more than usual, thus increasing the document's score despite the expression 'tottenham win' or 'tottenham victory' not appearing explicitly (within the defined slop). Notice that both systems, despite the difference in games returned and their positions, placed non-relevant documents with the same rankings. Therefore, both P@10 and AP assume equal (and inferior to the previous information need's) values across both.

Information need: Tottenham Hotspur home games that mention Heung-Min Son, despite him not scoring

Table 7: 'spurs AND (win OR victory)' query top 10 results.

	System 1		System 2	
Rank	**Game**	**R**	**Game**	**R**
1	West Brom 1-0 Tottenham 17/18	N	West Brom 1-0 Tottenham 17/18	N
2	Cardiff 0-3 Tottenham 18/19	Y	Cardiff 0-3 Tottenham 18/19	Y
3	Tottenham 3-1 Fulham 18/19	Y	Tottenham 3-1 Fulham 18/19	Y
4	Tottenham 5-0 Bournemouth 18/19	Y	Tottenham 5-0 Bournemouth 18/19	Y
5	Stoke 1-2 Tottenham 17/18	Y	Stoke 1-2 Tottenham 17/18	Y
6	Watford 2-1 Tottenham 18/19	N	Southampton 2-1 Tottenham 18/19	N
7	West Ham 0-1 Tottenham 18/19	Y	West Ham 0-1 Tottenham 18/19	Y
8	Tottenham 2-0 Watford 17/18	Y	Tottenham 2-0 Huddersfield 18/19	Y
9	Southampton 2-1 Tottenham 18/19	N	Watford 2-1 Tottenham 18/19	N
10	Bournemouth 1-4 Tottenham 17/18	Y	Palace 0-1 Tottenham 17/18	Y
P@10	0.7		0.7	
AP	0.6973		0.6973	

With this information need, the objective is to retrieve Tottenham games where the player Son did not feature in the goal scorer list, but was nonetheless mentioned in the game report. Ideally, he will have featured in the retrieved game list, and this will be reflected in the query parameters below. These kinds of needs (i.e., for the lack of a particular type of event instead of the presence of one) allows for the usage of more advanced search operators instead of just matching text to a number of fields.

The query chosen is [*son heung-min white hart lane -"son score"~5*]. What stands out in this query is the NOT operator, represented by a hyphen. It deems non-relevant all documents that contain the expression "son score" with a slop of at most 5 tokens. Some experiments were made to better adjust this offset value in order to both guarantee a wider coverage of the expression, while also not incorrectly matching wrong expressions.

Advanced search operators are what is mostly explored, feature wise, with this information need. There is also a usage of synonyms. Tottenham's old stadium, White Hart Lane, has suffered a renovation and has undergone a new name, Tottenham Hotspur Stadium. The synonym filter will ensure that 'white hart lane' matches documents in either of these venues. Tables 8 and 9 summarize the configuration and results for this query, respectively. Note that we're not searching against the home_scorers field, to try and avoid games where he scored.

The complete result set had 218 documents. Results are the same across systems and they achieved a perfect score under our evaluation metrics. A first glance at these results would maybe lead to think the query or need was too trivial, but we beg to differ and, instead, argue that the advanced search operators were the architect of the retrieval success. Their specificity and restriction help cut out otherwise possible results that we know (or estimate) will not be relevant, hence this. Not all search engines possess advanced search capabilities, however, even when present, are perhaps deemed too extensive or complex to remember (e.g., Google had over 40 search operators as of 2018 [21]), despite their aid potential.

Table 8: 'son heung-min white hart lane -"son score"~5' query parameter configuration.

	System 1	System 2
Parameters	qf=home_lineup home_subs report	qf=home_lineup^5 home_subs^5 report^10

Table 9: 'son heung-min white hart lane -"son score"~5' query retrieved results.

	System 1		System 2	
Rank	**Game**	**R**	**Game**	**R**
1	Tottenham 1-0 Sunderland 16/17	Y	Tottenham 1-0 Sunderland 16/17	Y
2	Tottenham 4-1 West Ham 15/16	Y	Tottenham 4-1 West Ham 15/16	Y
3	Tottenham 0-0 Chelsea 15/16	Y	Tottenham 0-0 Chelsea 15/16	Y
4	Tottenham 2-1 Man United 16/17	Y	Tottenham 2-1 Man United	Y
5	Tottenham 1-1 Burnley 17/18	Y	Tottenham 1-1 Burnley 17/18	Y
6	Tottenham 2-0 Man City 16/17	Y	Tottenham 2-0 Man City 16/17	Y
7	Tottenham 2-1 Southampton 16/17	Y	Tottenham 2-1 Southampton 16/17	Y
8	Tottenham 1-0 Middlesbrough 16/17	Y	Tottenham 1-0 Middlesbrough 16/17	Y
9	Tottenham 0-0 Swansea 17/18	Y	Tottenham 0-0 Swansea 17/18	Y
10	Tottenham 1-0 Southampton 14/15	Y	Tottenham 1-0 Southampton 14/15	Y
P@10	1		1	
AP	1		1	

Information need: Liverpool home games that had a goal from Mohamed Salah

With this information need, we intend to obtain all games played by Liverpool at their stadium and that featured at least one goal from one of their players, Mohamed Salah. Preferably, we will want to retrieve more recent games first, and thus will present new, query independent boosting mechanisms (aside from field boosting) that shall try to fulfill this particularity.

The query chosen is [*mohamed salah scores kop*]. There is nothing in particular that distinguishes this query from previously presented ones.

Feature wise, we will once again explore the usage of synonyms in the arena field (*kop* is a synonym for Anfield; see Listing 1). Regarding boost, we will use a boosting mechanism that increases the matching documents' score linearly (i.e., multiplies matching

score with boost sub-query score). We want to retrieve more recent matches first, so the boost factor will be 1/N, with N being the difference in time between the query date and the game date. In other words, more recent games will have a smaller difference, thus a bigger boosting factor. Tables 10 and 11 summarize the configuration and results for this query, respectively.

Table 10: 'mohamed salah scores kop' query parameter configuration.

	System 1	System 2
Parameters	qf=home_lineup home_subs home_scorers arena report	qf=home_lineup home_subs home_scorers^10 arena^10 report bf=div(1, ms(NOW/DAY,date))

Table 11: 'mohamed salah scores kop' query retrieved results.

	System 1		System 2	
Rank	**Game**	**R**	**Game**	**R**
1	Liverpool 4-0 Newcastle 18/19	Y	Liverpool 2-0 Chelsea 18/19	Y
2	Liverpool 5-0 Huddersfield 18/19	Y	Liverpool 5-0 Huddersfield 18/19	Y
3	Liverpool 2-2 Tottenham 17/18	Y	Liverpool 3-0 Bournemouth 18/19	Y
4	Liverpool 4-1 Cardiff 18/19	Y	Liverpool 1-0 Brighton 18/19	Y
5	Liverpool 2-1 Leicester 17/18	Y	Liverpool 4-3 Palace 18/19	Y
6	Liverpool 4-0 Brighton 17/18	Y	Liverpool 2-0 Fulham 18/19	Y
7	Liverpool 3-0 Bournemouth 17/18	Y	Liverpool 2-0 Wolves 18/19	N
8	Liverpool 2-1 Tottenham 18/19	N	Liverpool 5-1 Arsenal 18/19	Y
9	Liverpool 4-1 West Ham 17/18	Y	Liverpool 3-0 Southampton 18/19	Y
10	Liverpool 3-0 Huddersfield 17/18	N	Liverpool 4-0 Newcastle 18/19	Y
P@10	0.8		0.9	
AP	0.9861		0.9627	

The complete result set had 1520 documents, which represents 80% of the dataset. The reason for this amount is probably the amount of terms in the query that are treated as a union by default (i.e., like they were separated by OR). This indicates that the query probably wouldn't have performed well under higher recall levels. However, results show that both systems were capable of deciding relevance for the top 10 documents, which we think better suits a typical user's needs better here (a couple of good results near the top). In our efforts to boost more recent games, we ended up increasing precision, however, a non-relevant result ended up ranking higher, decreasing System 2's AP. It should be noted that some games, while not fully satisfying the information need, are not realistically far off and, therefore, their relevance assessment by the system can be considered expected (e.g., Liverpool 2-1 Tottenham was decided by a last minute Salah shot that was ultimately given as an own goal).

Information need: Games that mention Wolverhampton Wanderers, even if they're not directly involved

With this information need, the goal is to obtain all games in which Wolverhampton is mentioned, either by direct participation in it, or some other influence they might have had over it (e.g., they are one of the teams' next opponent, or recently played them).

The query chosen is [*wolve**]. In this query we will make use of wildcard search that allows matching multiple tokens at once (in this case, Wolves or Wolverhampton).

Feature wise, wildcard search is the focus of this information need. We could have used a synonym configuration for the team's name, as we did for others previously. However, for feature demonstration sake, we will not consider the existence of such synonym for this case. Wildcard search is also a more suitable alternative when searching for names or expressions on which we are unsure of their spelling, hence we find it relevant to show here. Tables 12 and 13 summarize the configuration and results for this query, respectively.

Table 12: 'wolve*' query parameter configuration.

	System 1	System 2
Parameters	qf=home away report	qf=home^5 away^5 report^10

The complete result set had 42 documents. Wolves, in the scope covered by this work, only participated in the league in 2018/2019, hence the majority of results in that time span. In a season there are 38 matches, therefore there are 4 matches in the dataset that mention Wolves even when they're not involved, and they were all captured by the second system. As expected, the games in question involve teams that had just played, were going to play or were in a similar league position at the time as Wolves (e.g., Arsenal had just lost to them prior to playing Leicester).

Results achieved perfect precision at 10 and average precision once again. The information need and query were simple to interpret and represent, therefore these values were expected.

3.5 Tool evaluation

Solr is a good tool for an information retrieval project. The documentation is thorough and provides some out of the box examples that help anyone quickly setup an instance and explore their functionalities. Indexing our dataset was simple and allowed us to explore various tokenizers and filters available.

Not all tasks presented the same difficulty. In particular, the third task presented (Tottenham Hotspur home games that mention Heung-Min Son, despite him not scoring) involved some experimenting to achieve an expression (and slop value) that would lead to good results. Perhaps the biggest challenge was using boost

Table 13: 'wolve*' query retrieved results.

	System 1		System 2	
Rank	**Game**	**R**	**Game**	**R**
1	Palace 1-2 Bournemouth 15/16	Y	Palace 1-2 Bournemouth 15/16	Y
2	Liverpool 2-0 Wolves 18/19	Y	Leicester 3-0 Arsenal 18/19	Y
3	Wolves 1-0 Fulham 18/19	Y	Man United 1-1 Chelsea 18/19	Y
4	Leicester 3-0 Arsenal 18/19	Y	Wolves 3-1 Arsenal 18/19	Y
5	Man United 1-1 Chelsea 18/19	Y	Burnley 2-0 Wolves 18/19	Y
6	Watford 1-2 Wolves 18/19	Y	Chelsea 1-1 Wolves 18/19	Y
7	Wolves 3-1 Arsenal 18/19	Y	Wolves 1-1 Newcastle 18/19	Y
8	Wolves 0-0 Brighton 18/19	Y	Chelsea 2-0 Fulham 18/19	Y
9	Southampton 3-1 Wolves 18/19	Y	Wolves 1-0 Burnley 18/19	Y
10	Wolves 2-1 Man United 18/19	Y	Liverpool 2-0 Wolves 18/19	Y
P@10	1		1	
AP	1		1	

functions, as done in the fourth task presented (Liverpool home games with a goal from Mohamed Salah). Solr's documentation still mentioned Negative Boosts, despite their now apparent lack of support, and we spent some time trying to make them work before finally moving to an alternative, inverted positive boost.

Results overall were satisfactory. Precision was never lower than 70% and Average Precision had a similar lower bound. Higher precision levels in the top results allow a user to quickly obtain some relevant documents, which should fit a typical user's profile in this context. While we understand that the set of information needs presented is not enough for a thorough system evaluation, it is interesting to note that systems had similar performances throughout testing. Average precision values are more similar in the two systems for a single information need than for multiple needs in one system. This goes in accordance to what is said by Manning et al. [24].

4 SEMANTIC WEB

The World Wide Web (WWW), since its origin, quickly became the leading platform for sharing information, initially in the form of documents. As time went on, the demand shifted, as people needed to share more and more data of different types. This brought a series of automation issues, as machines could not understand pages and documents beyond their syntactical structure, as humans did, unless explicitly told how to. This is what the Semantic Web addresses, by proposing a set of technologies whose ultimate goal was for information spread in the web to be linked so that different pages could interact without specific protocols that would not work outside that particular communication, as well as enrich them with machine-interpretable metadata. Data is described as a set of entities that together form vocabularies, or ontologies, with a set of data properties that characterize them and object properties that establish relationships in a graph structure.

In his original proposal [3], Sir Tim Berners-Lee saw the Semantic Web as an enabler for computers to handle information like humans, i.e., understand its meaning and relation with other existing information, without the need for their manual intervention. This idea is what coined the "semantic" term in Semantic Web.

In this work, we look at our domain from this new perspective and try to identify what entities exist and how they can be defined in a way that can be easily parsed by machines and integrated with existing vocabularies.

4.1 Data domain and concepts

The previously presented data conceptual model mostly contains the main entities that are used in this work (see Figure 1). However, for clarity and further semantic enrichment of our domain, we decided to include, at this stage, a new concept: an Appearance. This concept describes an appearance by a football player in a specific game and is characterized by the team he represented at that time, as well as the different highlights of that game associated to the same player (if applicable). We believe this addition will make our ontology more meaningful, as appearances help group all highlights of a player in order to more easily define their exhibition in a match, and this milestone presents the best opportunity to introduce it.

4.2 Existing ontologies

Football is the most popular in the world [11] and, thus, much information is shared about it. With this in mind, many ontologies have been defined that, in a way or another, try to model concepts associated with this particular sport or with more generic sporting events. Though our focus is football, we also looked at ontologies that were at a higher abstraction level, but nevertheless presented interesting concepts.

Many projects have a limited scope or are defined in a very isolated way. FootballNET [22] is an ontology that focuses on modelling club data (name, city and past accomplishments, though the latter is simply a string whose content resembles that which is present in the clubs' Wikipedia page) and, as of the time of writing, includes partial information on a small subset of clubs from the English Premier League. Likewise, RDF OWL Soccer [25] is the result of a small academic project that targeted the top four Premier League clubs in the 2012/2013 season. The DAML Soccer Ontology [10] is an older project that only appears to define higher-level concepts (e.g., clubs and competitions) and does not integrate with other vocabularies. All of these were deemed too limiting in terms of the concepts/data pair that they offer.

Other ontologies are not specifically aimed at football, however include related concepts in their vocabulary. The BBC Sports Ontology [2] is a project by the news platform that is the result of an extensive study on event sharing (relying on good work done previously) applied to the sports domain. Though well defined, it is somewhat too generic. Proton (PROTo ONtology) [30] is a project that aims at being a seed for the generation of specialized ontologies and already has a number of interesting concepts for our domain (e.g., club, player, match). Schema.org [35], a more

well-known project, also aims at developing standard schemas for structured information on the Internet and is used widely. However, our ontology choice, in the end, was the DBpedia Ontology [13]. A cross-domain project, it contains several entities, many that we use in our domain (e.g., Club, Player, Match, Stadium). It also links with a series of other ontologies (e.g., Schema.org) and external datasets, such as a dump from the YAGO project [44], a semantic knowledge base derived from sources such as Wikipedia [49], which presents us with the opportunity of not only integrating our domain model with existing vocabulary, but also expanding the information provided beyond our dataset. The biggest drawback is the lack of existing vocabulary for game events. Also, not all properties from our domain are included in the original ontology. These factors, though, provide an innovative component to the extension that will be our new ontology.

4.3 Ontology creation

To create our ontology, we used the Protégé [31] tool, which provides an easy to use interface, presents reasonable online documentation and support and can be extended through plugins to, for example, visualize the ontology as a graph or query it.

Before defining our ontology, we created a set of prefixes that help us link to existing concepts and properties that we can reuse, so as to avoid repetition of similar resources that really mean the same thing. More specifically, we mapped the prefixes dbo and dbp to a selected subset of classes and properties from the DBpedia ontology [13], as well as the foaf prefix to the Friend of a Friend (FOAF) ontology [18].

In terms of classes, we were able to directly use various existing entities from the DBpedia ontology, more specifically *FootballMatch*, *Referee*, *SoccerClub*, *SoccerLeagueSeason*, *SoccerPlayer* and *Stadium*. Figure 6 presents an overview of all classes.

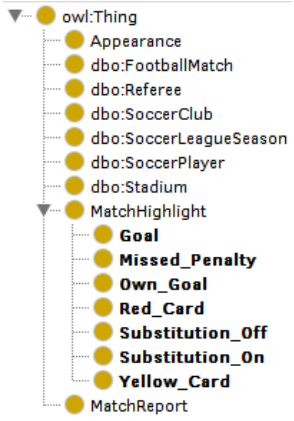

Figure 6: Ontology classes.

Though the naming inconsistency (Football vs Soccer) is somewhat unsettling, these were effectively the classes used in existing DBpedia resources, so we stood by them. The concepts that our ontology introduces as new are related mostly with game highlights: the *Appearance*, as described in our domain analysis, and *MatchHighlight* classes, with all subclasses of the latter being disjoint between them. There is also the concept of the *MatchReport* to store the natural language, textual description of the match.

Regarding object properties, we had to resort mostly to our own definitions. However, we were still able to reuse the properties of a match's referee and the arena where it occurred. All object properties are shown in Figure 7.

Figure 7: Ontology object properties.

Most of the object properties had their inverse defined as well, which meant inverted characteristics too. Therefore, about half of them are functional, while the other half are inverse functional (other properties, such as symmetry and transitiveness, do not apply in the domain). The only exception is the *matchReport* property, which maps a *FootballMatch* to a *MatchReport*, that is functional and has no inverse, since it is a one to one association. The *player* property (functional) maps an *Appearance* to a *SoccerPlayer* and has as inverse the *appearances* property (inverse functional). Similarly, *highlightAppearance* maps *MatchHighlight* to *Appearance*, with *highlight* as inverse. The same can be said for the remaining functional/inverse functional object property pairs *homeTeam*/*homeGames*, *awayTeam*/*awayGames*, *referee*/*refereedGames*, *stadium*/*stadiumGames*, *season*/*seasonGames*, *team*/*teamAppearances* (linking *Appearance* and *SoccerClub*) and *match*/*exhibition* (linking *Appearance* with *FootballMatch*).

Finally, in the data properties, we were able to, once again, integrate with existing properties. The FOAF *name* property is multipurpose (i.e., used for stadium, referee, club and player names), while attendance and date are properties already used in the context of the DBpedia ontology. All score properties refer to a match, the *endingYear* defines a season, the *minute* will be used to pinpoint when a highlight occurred and the pair *title*/*content* will be used for match reports, with the possibility of publishing both properties in different languages (e.g., title@en = 'Football game summary' and title@pt = 'Resumo de jogo de futebol'). Though we do not have titles in our dataset, it seemed natural to structure the entity this way. In the future, other expansions could be made (e.g.,

an object property mapping to the report's author). All properties are functional and were defined as such. Figure 8 presents all data properties.

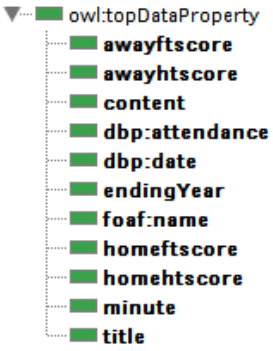

Figure 8: Ontology data properties.

4.4 Ontology population

Populating the ontology required us to take a few steps back and, once again, rethink the structure of our dataset. The reason for this is all (semi-)automatic ontology populating tools/plugins, such as OntoRefine [27] and Cellfie [7], only work with tabular data, and our dataset is mostly a hierarchical JSON. We initially considered coding a parser for our specific case ourselves, but ultimately deemed the effort not worthy. Therefore, the first task was to split our dataset into multiple, tabular datasets. In the end, we obtained three, CSV formatted sub-datasets: game information (each line storing a game with information on season, teams, date, scores, match report and referee), highlight appearances (each row maps to a highlight of a player in a game) and pure appearances (mapping players to games where they played, but had no highlights as defined by us previously). Appendix C contains example entries for all of them.

With appropriate datasets, we move to the population process itself. For it, we opted to use the Cellfie plugin which is already integrated with Protégé and allows to process spreadsheets with an extension of the OWL2 Manchester Syntax [45] that provides cell referencing. By defining a set of rules to create entities from each class previously defined and their set of properties (see Figure 9), we were able to parse our information and load it into the ontology, where it can be queried. We annex a JSON file (*transformation_rules.json*) describing the full transformation rule set used to parse and populate the ontology. Protégé is a resource-heavy tool, both in terms of CPU and memory usage (e.g., workloads such as dataset importing or querying averaged >=80% of usage in the single core it ran, as well as over 2GB of memory). Hence, we only loaded a fifth of our dataset, corresponding to the 2014/2015 season.

4.5 Querying

We now focus on answering the proposed retrieval tasks that were not answered in the Information Retrieval section. Some tasks had to be readjusted, as they target either specific seasons that we did not load or even all seasons. We point out the modifications as we answer each one. For each task, we reiterate the associated question,

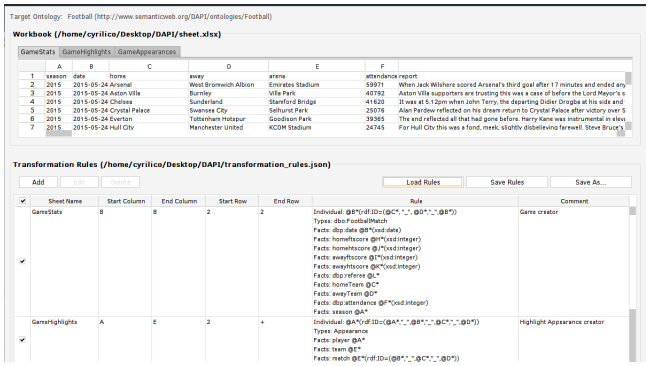

Figure 9: Cellfie overview with a subset of transformation rules visible.

necessary modifications compared to their original proposal, the SPARQL [46] query that answers it and the results obtained.

Question: Which team wins more often with Mike Dean as the referee?

Here, we try to see what team has the most victories, either home or away, when a game is officiated by a specific referee. Listing 2 shows the query that answers it. Notice the usage of the UNION operator, as we want to merge two possible paths (home and away wins). Before splitting paths, we retrieve games officiated by the wanted referee, and then check if the corresponding team obtained a bigger score than their opposition by using the FILTER operator. Unfortunately, score predicates had to be duplicated inside each branch in order for the FILTER operator to work properly. Figure 10 shows the query's results.

Listing 2: "Which team wins more often with Mike Dean as the referee?" SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-
    rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX fo: <http://www.semanticweb.org/DAPI/
    ontologies/Football/>

SELECT ?teamname (count(?game) as ?games)
WHERE {
?team rdf:type dbo:SoccerClub .
?team foaf:name ?teamname .
?game dbp:referee ?ref .
?ref foaf:name "M␣Dean"^^xsd:string .
{
?game fo:homeftscore ?scoreHome .
?game fo:awayftscore ?scoreAway .
 ?game fo:homeTeam ?team . FILTER(?scoreHome
      > ?scoreAway) }
UNION
{
```

```
?game fo:homeftscore ?scoreHome .
?game fo:awayftscore ?scoreAway .
?game fo:awayTeam ?team . FILTER(?scoreAway
    > ?scoreHome) }
}
GROUP BY ?teamname
ORDER BY DESC(?games)
LIMIT 1
```

teamname	games
"Chelsea"	"2"^^<http://www.w3.org/2001/XMLSchema#integ

Figure 10: Results for "Which team wins more often with Mike Dean as the referee?".

Question: Which teams are most carded during second halves?

Here, we assess which teams receive most yellow and red cards during the second 45 minutes of matches. Notice that, compared to the original proposal, we now do not limit the result to one team only (the most carded), but instead want a sorted output. Listing 3 shows the query that answers it. We make usage of the UNION operator once again, as we need to consider multiple paths (yellow and red cards). We filter for highlights with a minute superior to 45, as that is what describes being part of the second half (minutes 46 to 90). Figure 11 shows the corresponding result.

Listing 3: "Which teams are most carded during second halves?" SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-
    rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX fo: <http://www.semanticweb.org/DAPI/
    ontologies/Football/>

SELECT ?teamname (count(?highlight) as ?
    cards)
WHERE {
?team rdf:type dbo:SoccerClub .
?team foaf:name ?teamname .
?app fo:team ?team .
?highlight fo:highlightAppearance ?app .
{?highlight rdf:type fo:Yellow_Card}
UNION
{?highlight rdf:type fo:Red_Card}
?highlight fo:minute ?min .
FILTER (?min > 45)
}
GROUP BY ?teamname
ORDER BY DESC(?cards)
```

Question: What is the most common substitution in Tottenham Hotspur?

teamname	cards
"Sunderland"	"67"^^<http://www.w3.org/2001/XMLSchema#inte
"Aston Villa"	"60"^^<http://www.w3.org/2001/XMLSchema#inte
"Newcastle United"	"57"^^<http://www.w3.org/2001/XMLSchema#inte
"Hull City"	"56"^^<http://www.w3.org/2001/XMLSchema#inte
"Tottenham Hotspur"	"56"^^<http://www.w3.org/2001/XMLSchema#inte
"Manchester United"	"53"^^<http://www.w3.org/2001/XMLSchema#inte
"Queens Park Rangers"	"52"^^<http://www.w3.org/2001/XMLSchema#inte
"Chelsea"	"52"^^<http://www.w3.org/2001/XMLSchema#inte
"Manchester City"	"52"^^<http://www.w3.org/2001/XMLSchema#inte
"Arsenal"	"51"^^<http://www.w3.org/2001/XMLSchema#inte
"Stoke City"	"50"^^<http://www.w3.org/2001/XMLSchema#inte
"Crystal Palace"	"49"^^<http://www.w3.org/2001/XMLSchema#inte
"Everton"	"49"^^<http://www.w3.org/2001/XMLSchema#inte
"Burnley"	"47"^^<http://www.w3.org/2001/XMLSchema#inte
"Liverpool"	"45"^^<http://www.w3.org/2001/XMLSchema#inte
"West Ham United"	"43"^^<http://www.w3.org/2001/XMLSchema#inte
"West Bromwich Albion"	"43"^^<http://www.w3.org/2001/XMLSchema#inte
"Southampton"	"40"^^<http://www.w3.org/2001/XMLSchema#inte
"Leicester City"	"37"^^<http://www.w3.org/2001/XMLSchema#inte
"Swansea City"	"37"^^<http://www.w3.org/2001/XMLSchema#inte

Figure 11: Results for "Which teams are most carded during second halves?".

Here, we assess common player changes in matches for a particular team. We do not explicitly link substitution off and substitution on highlights, whether it is in our dataset, or our ontology, however they can be matched to each other if they occurred in the same minute. Notice that, compared to the original proposal, we eliminated the 2017/2018 season filter. Listing 4 shows the query used to answer it. We start by retrieving all Tottenham matches, then all appearances in that match, and finally pair appearances with substitution on and substitution off highlights in the same minute. Figure 12 shows the answer to the query.

Listing 4: "What is the most common substitution in Tottenham Hotspur?" SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-
    rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX fo: <http://www.semanticweb.org/DAPI/
    ontologies/Football/>

SELECT ?outname ?inname (count(*) as ?nTimes
    )
WHERE {
{?game fo:homeTeam ?team .
?team foaf:name "Tottenham␣Hotspur"}
UNION
{?game fo:awayTeam ?team .
?team foaf:name "Tottenham␣Hotspur"}
?appOut fo:match ?game .
?appIn fo:match ?game .
?appOut fo:team ?team .
?appIn fo:team ?team .
?highlightOut fo:highlightAppearance ?appOut
    .
?highlightIn fo:highlightAppearance ?appIn .
?highlightOut rdf:type fo:Substitution_Off .
?highlightIn rdf:type fo:Substitution_On .
```

```
?highlightOut fo:minute ?min .
?highlightIn fo:minute ?min .
?appOut fo:player ?playerOut .
?appIn fo:player ?playerIn .
?playerOut foaf:name ?outname .
?playerIn foaf:name ?inname
}
GROUP BY ?outname ?inname
ORDER BY DESC(?nTimes)
```

outname	inname	nTimes
"Ryan Mason"	"Paulinho"	"5"^^<http://www.w3.org/2001/XI
"Nacer Chadli"	"Roberto Soldado"	"5"^^<http://www.w3.org/2001/XI
"Andros Townsend"	"Mousa Dembélé"	"5"^^<http://www.w3.org/2001/XI
"Harry Kane"	"Roberto Soldado"	"4"^^<http://www.w3.org/2001/XI
"Christian Eriksen"	"Mousa Dembélé"	"4"^^<http://www.w3.org/2001/XI
"Ryan Mason"	"Benjamin Stambouli"	"4"^^<http://www.w3.org/2001/XI
"Danny Rose"	"Ben Davies"	"4"^^<http://www.w3.org/2001/XI
"Andros Townsend"	"Erik Lamela"	"3"^^<http://www.w3.org/2001/XI
"Nacer Chadli"	"Erik Lamela"	"3"^^<http://www.w3.org/2001/XI
"Nacer Chadli"	"Mousa Dembélé"	"3"^^<http://www.w3.org/2001/XI
"Emmanuel Adebayor"	"Harry Kane"	"3"^^<http://www.w3.org/2001/XI
"Ryan Mason"	"Nacer Chadli"	"2"^^<http://www.w3.org/2001/XI
"Mousa Dembélé"	"Nacer Chadli"	"2"^^<http://www.w3.org/2001/XI
"Christian Eriksen"	"Erik Lamela"	"2"^^<http://www.w3.org/2001/XI
"Ryan Mason"	"Mousa Dembélé"	"2"^^<http://www.w3.org/2001/XI
"Andros Townsend"	"Paulinho"	"2"^^<http://www.w3.org/2001/XI
"Nabil Bentaleb"	"Mousa Dembélé"	"2"^^<http://www.w3.org/2001/XI
"Emmanuel Adebayor"	"Roberto Soldado"	"2"^^<http://www.w3.org/2001/XI
"Nacer Chadli"	"Paulinho"	"2"^^<http://www.w3.org/2001/XI
"Erik Lamela"	"Nacer Chadli"	"2"^^<http://www.w3.org/2001/XI

Figure 12: Results for "What is the most common substitution in Tottenham Hotspur?".

Question: Which players score at least in seven different months?

Here, we assess consistency in goal scoring form by checking what players score in different periods throughout the season. Since the original proposal, we removed the filter that this would need to apply to all seasons. Since we only have one, we will focus on it. We changed the restriction from five to seven months in order to shorten our output, which could otherwise be considerably large. Listing 5 shows the query that answers the question. We start by grouping distinct pairs player/scoring month, as Protégé did not seem to directly support a COUNT(DISTINCT...) clause (it would perform a normal COUNT). To extract the scoring month, we extract this date portion using regular expressions and bind the result to a new variable. On our main query, we then regroup, this time by player, and filter through the HAVING clause.

Unfortunately, we were unable to run this query using the available Protégé SPARQL plugins. The default plugin does not recognize operators such as REPLACE (used to extract the month; other alternatives included a MONTH function, which did not work either). This is because the plugin is still running an older SPARQL API [6, 36], with there being little motivation to upgrade by the maintainers [4]. Alternatives that support these functions exist, like the SNAP SPARQL plugin [42], but this plugin does not support subqueries [9], nor the HAVING modifier [5].

Therefore, we ended up using Apache Jena [1] just for this query, since it was the last we implemented. This tool provides some helper scripts to run queries against ontologies. The results are shown in Figure 13.

Listing 5: "Which players score at least in seven different months?" SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-
    rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX fo: <http://www.semanticweb.org/DAPI/
    ontologies/Football/>

SELECT ?playername (count(?goalMonth) as ?
    nMonths)
WHERE {
{
SELECT DISTINCT ?playername ?goalMonth
WHERE {
?app fo:match ?game .
?app fo:player ?player .
?player foaf:name ?playername .
?highlight fo:highlightAppearance ?app .
?highlight rdf:type fo:Goal .
?game dbp:date ?date .
BIND(replace(?date, "^....-(..)-..$", "$1")
    AS ?goalMonth)
}
}
}
GROUP BY ?playername
HAVING(?nMonths >= 7)
ORDER BY DESC(?nMonths)
```

playername	nMonths
"Charlie Austin"	10
"Alexis Sánchez"	9
"Eden Hazard"	9
"Diego Costa"	8
"Mame Biram Diouf"	8
"Sergio Agüero"	8
"David Silva"	7
"Harry Kane"	7
"Juan Mata"	7
"Nacer Chadli"	7
"Olivier Giroud"	7
"Romelu Lukaku"	7
"Saido Berahino"	7
"Wayne Rooney"	7

Figure 13: Results for "Which players score at least in seven different months?".

Question: What is the percentage of goals scored in the last 15 minutes of matches?

Here, we assess how often teams score late in matches. Originally, we wanted to see which season had the highest percentage, however, here, we will focus on the value for the season we have at

hand. Listing 6 shows the answering query. For the sake of demonstration, and to show potential usage of the *Season* entity, both versions, considering only the populated ontology and imagining we had all data loaded, are presented, though we only show results for the former, in Figure 14.

Listing 6: "What is the percentage of goals scored in the last 15 minutes of matches?" SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-
    rdf-syntax-ns#>
PREFIX fo: <http://www.semanticweb.org/DAPI/
    ontologies/Football/>

#Version considering populated ontology
SELECT (sum(?last15)/count(*) as ?
    last15goalratio)
WHERE {
?app fo:match ?game .
?highlight fo:highlightAppearance ?app .
{?highlight rdf:type fo:Goal} UNION {?
    highlight rdf:type fo:Own_Goal} .
?highlight fo:minute ?minute .
BIND(IF(?minute > 75, 1, 0) as ?last15)
}

#Version considering full population loaded
    (i.e., grouped by season)
SELECT ?season (sum(?last15)/count(*) as ?
    last15goalratio)
WHERE {
?game fo:season ?season .
?app fo:match ?game .
?highlight fo:highlightAppearance ?app .
{?highlight rdf:type fo:Goal} UNION {?
    highlight rdf:type fo:Own_Goal} .
?highlight fo:minute ?minute .
BIND(IF(?minute > 75, 1, 0) as ?last15)
}
GROUP BY ?season
ORDER BY DESC(?last15goalratio)
LIMIT 1
```

last15goalratio
"0.21047227926078028747433"^^<http://www.w3.org/2001/XMLSchema#decimal>

Figure 14: Results for "What is the percentage of goals scored in the last 15 minutes of matches?".

Question: Which player scored more goals at Old Trafford?

Here, we assess the most prolific players in a given arena. We decided to consider own goals as a prolific moment as well, hence their inclusion in the query show in Listing 7. Figure 15 shows the respective results.

Listing 7: "Which player scored more goals at Old Trafford?" SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-
    rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/
    XMLSchema#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX fo: <http://www.semanticweb.org/DAPI/
    ontologies/Football/>

SELECT ?playername (count(?highlight) as ?
    goals)
        WHERE {
?game dbp:stadium ?arena .
?arena foaf:name "Old Trafford"^^xsd:string
    .
?app fo:match ?game .
?app fo:player ?player .
?player foaf:name ?playername .
?highlight fo:highlightAppearance ?app .
{?highlight rdf:type fo:Goal}
UNION
{?highlight rdf:type fo:Own_Goal}
}
GROUP BY ?playername
ORDER BY DESC(?goals)
```

playername	goals
"Wayne Rooney"	"11"^^<http://www.w3.org/2001/XMLSchema#integer>
"Robin van Persie"	"7"^^<http://www.w3.org/2001/XMLSchema#integer>
"Juan Mata"	"5"^^<http://www.w3.org/2001/XMLSchema#integer>
"Ander Herrera"	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
"Chris Smalling"	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
"Marouane Fellaini"	"3"^^<http://www.w3.org/2001/XMLSchema#integer>
"Angel di Maria"	"2"^^<http://www.w3.org/2001/XMLSchema#integer>
"Sergio Agüero"	"2"^^<http://www.w3.org/2001/XMLSchema#integer>
"Falcao"	"2"^^<http://www.w3.org/2001/XMLSchema#integer>
"Tyler Blackett"	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
"Steven N'Zonzi"	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
"Papiss Cisse"	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
"Dusan Tadic"	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
"Christian Benteke"	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
"Michael Carrick"	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
"Diafra Sakho"	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
"Danny Ings"	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
"Didier Drogba"	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
"Sung-Yueng Ki"	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
"Gylfi Sigurdsson"	"1"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 15: Results for "Which player scored more goals at Old Trafford?".

Question: What players score the most under big crowds (e.g., games with above 50000 attendance)?

Here, take another perspective on the prolific filter, as we try to assess players that keep a good record even in moments of potentially bigger pressure. In our original proposal, we set 50000 as a threshold, as audiences usually lie within the tens of thousands. We decided to keep it and use that value as our minimum. Listing 8 shows the answering query. Figure 16 contains the result.

Listing 8: "What players score the most under big crowds?" SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-
    rdf-syntax-ns#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

```
PREFIX fo: <http://www.semanticweb.org/DAPI/
    ontologies/Football/>

SELECT ?playername (count(?highlight) as ?
    goals)
        WHERE {
?game dbp:attendance ?attendance .
FILTER(?attendance > 50000) .
?app fo:match ?game .
?app fo:player ?player .
?player foaf:name ?playername .
?highlight fo:highlightAppearance ?app .
{?highlight rdf:type fo:Goal}
UNION
{?highlight rdf:type fo:Own_Goal}
}
GROUP BY ?playername
ORDER BY DESC(?goals)
```

playername	goals
"Wayne Rooney"	"12"^^<http://www.w3.org/2001/XMLSchema#integer>
"Alexis Sánchez"	"9"^^<http://www.w3.org/2001/XMLSchema#integer>
"Olivier Giroud"	"9"^^<http://www.w3.org/2001/XMLSchema#integer>
"Robin van Persie"	"7"^^<http://www.w3.org/2001/XMLSchema#integer>
"Theo Walcott"	"5"^^<http://www.w3.org/2001/XMLSchema#integer>
"Papiss Cisse"	"5"^^<http://www.w3.org/2001/XMLSchema#integer>
"Juan Mata"	"5"^^<http://www.w3.org/2001/XMLSchema#integer>
"Sergio Agüero"	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
"Ander Herrera"	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
"Moussa Sissoko"	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
"Chris Smalling"	"4"^^<http://www.w3.org/2001/XMLSchema#integer>
"Laurent Koscielny"	"3"^^<http://www.w3.org/2001/XMLSchema#integer>
"Santi Cazorla"	"3"^^<http://www.w3.org/2001/XMLSchema#integer>
"Ayoze Pérez"	"3"^^<http://www.w3.org/2001/XMLSchema#integer>
"Marouane Fellaini"	"3"^^<http://www.w3.org/2001/XMLSchema#integer>
"Tomas Rosicky"	"2"^^<http://www.w3.org/2001/XMLSchema#integer>
"Aaron Ramsey"	"2"^^<http://www.w3.org/2001/XMLSchema#integer>
"Didier Drogba"	"2"^^<http://www.w3.org/2001/XMLSchema#integer>
"Hector Bellerin"	"2"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 16: Results for "What players score the most under big crowds?".

Question: Which team let a lead slip the most at home (i.e., scored first, but didn't win the game)?

With this question, we want to see what teams can't hold an advantage often, i.e., the first goal in the game is theirs, but they ultimately do not win the game. From our original proposal, we scrapped the 2015/2016 season filter, as well as restricted this to teams playing at home only. While querying the same logic for away teams would likely be a matter of copy and paste, the query would become too big and lose its readability, which is important here since we are equally interested in result and query.

Listing 9 shows this query. We resorted to a subquery to obtain the team/game pairs where the condition is met. We apply an early FILTER that eliminates goalless games. We then calculate the match's first goal of any team. Our subquery ends with a UNION between two sub-blocks. The first checks the minutes of the goals by the home team, while the second checks the minutes for own goals by the away team (which is also a home team goal in the context). Finally, we filter the groupings to include only the matches where the home team did score the first goal. Then, in our main query, we just regroup, this time only by the team, and sort the result. Figure 17 shows the results for this query.

Listing 9: "Which team let a lead slip the most at home (i.e. scored first but didn't win the game)?" SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-
    rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX fo: <http://www.semanticweb.org/DAPI/
    ontologies/Football/>

SELECT ?team (count(?game) as ?games)
WHERE{
{
SELECT ?team ?game
WHERE {
?app fo:match ?game .
?game fo:homeftscore ?homeScore .
?game fo:awayftscore ?awayScore .
FILTER(?homeScore + ?awayScore > 0 && ?
    awayScore >= ?homeScore) .
?generalhighlight fo:highlightAppearance ?
    app .
{?generalhighlight rdf:type fo:Goal} UNION
    {?generalhighlight rdf:type fo:Own_Goal}
    .
?generalhighlight fo:minute ?generalminute .
{?homeapp fo:match ?game .
?homeapp fo:team ?team .
?game fo:homeTeam ?team .
?homehighlight fo:highlightAppearance ?
    homeapp .
?homehighlight rdf:type fo:Goal .
?homehighlight fo:minute ?homeminute}
UNION
{?awayapp fo:match ?game .
?awayapp fo:team ?away .
?game fo:awayTeam ?away .
?awayhighlight fo:highlightAppearance ?
    awayapp .
?awayhighlight rdf:type fo:Own_Goal .
?awayhighlight fo:minute ?homeminute}
}
GROUP BY ?team ?game
HAVING (min(?generalminute) = min(?
    homeminute))
}
}
GROUP BY ?team
ORDER BY DESC(?games)
LIMIT 1
```

Question: What is Tottenham Hotspur's win ratio when Harry Kane starts?

Now, we want to assess a player's importance in their team, by

team	games
Liverpool	"5"^^<http://www.w3.org/2001/XMLSchema#integer>

Figure 17: Results for "Which team let a lead slip the most at home (i.e., scored first, but didn't win the game)?".

checking Tottenham's win ratio when Harry Kane is in the starting lineup (that does not include appearances as a used substitute). From the original proposal, we removed another player that we would consider, Dele Alli, as he only arrived in 2015/2016. We also removed the season filter.

Listing 10 shows the respective query. First, we extract games in which Tottenham participated and add an auxiliary variable (result) that is equal to 1 if they won that game, and 0 otherwise. This is how we calculate the ratio in the end: we sum this value for each row (giving us a win count) and divide by the number of games. To filter by games that Kane started, we first apply a FILTER that assures there is an appearance by that player. The second FILTER guarantees that this appearance is not as a substitute. Figure 18 shows the query's results.

Listing 10: "What is Tottenham Hotspur's win ratio when Harry Kane starts?" SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-
    rdf-syntax-ns#>
PREFIX xsd: <http://www.w3.org/2001/
    XMLSchema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX fo: <http://www.semanticweb.org/DAPI/
    ontologies/Football/>

SELECT (sum(?result)/count(*) as ?winratio)
WHERE {
{?game fo:homeTeam ?team .
?team foaf:name "Tottenham Hotspur"^^xsd:
    string .
?game fo:homeftscore ?scoreHome .
?game fo:awayftscore ?scoreAway .
BIND(IF(?scoreHome > ?scoreAway, 1, 0) as ?
    result)}
UNION
{?game fo:awayTeam ?team .
?team foaf:name "Tottenham Hotspur"^^xsd:
    string .
?game fo:homeftscore ?scoreHome .
?game fo:awayftscore ?scoreAway .
BIND(IF(?scoreAway > ?scoreHome, 1, 0) as ?
    result)}
FILTER EXISTS {?app fo:match ?game ; fo:team
     ?team ; fo:player ?player . ?player
    foaf:name "Harry Kane"^^xsd:string}
FILTER NOT EXISTS {?app fo:match ?game ; fo:
    team ?team ; fo:player ?player . ?player
     foaf:name "Harry Kane"^^xsd:string . ?
    highlight fo:highlightAppearance ?app ;
    rdf:type fo:Substitution_On}
}
```

winratio
"0.535714285714285714285714"^^<http://www.w3.org/2001/XMLSchema#decimal>

Figure 18: Results for "What is Tottenham Hotspur's win ratio when Harry Kane starts?".

4.6 Integration with other ontologies

Our ontology can be linked to other existing ontologies in order to expand the information provided beyond our dataset. To briefly explore this, we exported, using the DBpedia SPARQL endpoint [12], a subset of their populated ontology with information regarding Premier League clubs and their properties. To try and ensure we only extracted information from English clubs, so as to not overload Protégé, we used classes defined by a YAGO dump, WikicatFootballClubsInEngland and WikicatPremierLeagueClubs. While one could have been enough, surprisingly, none is a subset of the other (and we would expect that the latter is a subset of the former, at least). Thus, we decided to unite both. Listing 11 shows the CONSTRUCT query that built the extracted triples.

Listing 11: DBpedia team information export SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-
    rdf-syntax-ns#>
PREFIX yago: <http://dbpedia.org/class/yago
    />

CONSTRUCT {?team ?property ?value}
WHERE {
{?team rdf:type yago:
    WikicatPremierLeagueClubs}
UNION
{?team rdf:type yago:
    WikicatFootballClubsInEngland} .
?team ?property ?value .
}
```

After loading this ontology subset, we could query both at the same time and link them. From this new ontology, we can query new information, such as the teams' stadiums or managers. Listings 12 and 13 show the queries used to obtain this information, while Figures 19 and 20 show their results, respectively. Both are similar, with the main difference being the last property in the WHERE clause. We first retrieve the teams stored in our ontology by filtering all teams that have their name with a datatype of string, as defined by the XML Schema [47] (the instances from the DBpedia ontology do not respect this condition). We then retrieve the remaining teams and merge them on their name. The usage of the STR function allows strings to be parsed to literal values without, for example, language annotations, that would render the comparison impossible otherwise.

Listing 12: Extract team stadiums SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-
    rdf-syntax-ns#>
```

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?teamname ?teamstadium
WHERE {
?teamInternal rdf:type dbo:SoccerClub .
?teamInternal foaf:name ?localName .
FILTER (datatype(?localName) = xsd:string) .
BIND(STR(?localName) as ?teamname).
?teamExternal rdf:type dbo:SoccerClub .
?teamExternal foaf:name ?externalName .
FILTER(STR(?externalName) = ?teamname) .
?teamExternal dbo:ground ?teamstadium
}
```

Listing 13: Extract team managers SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-
    rdf-syntax-ns#>
PREFIX yago: <http://dbpedia.org/class/yago
    />

SELECT ?teamname ?teammanager
WHERE {
?teamInternal rdf:type dbo:SoccerClub .
?teamInternal foaf:name ?localName .
FILTER (datatype(?localName) = xsd:string) .
BIND(STR(?localName) as ?teamname).
?teamExternal rdf:type dbo:SoccerClub .
?teamExternal foaf:name ?externalName .
FILTER(STR(?externalName) = ?teamname) .
?teamExternal dbo:manager ?teammanager
}
```

Figures 19 and 20 show the results for the queries, in the same order.

teamname	teamstadium
"Everton"	dbpedia:Goodison_Park
"Crystal Palace"	dbpedia:Selhurst_Park
"Tottenham Hotspur"	dbpedia:2016–17_Tottenham_Hotspur_F.C._season
"West Bromwich Albion"	dbpedia:The_Hawthorns
"Queens Park Rangers"	dbpedia:Loftus_Road
"Leicester City"	dbpedia:King_Power_Stadium
"Arsenal"	dbpedia:Emirates_Stadium
"Tottenham Hotspur"	dbpedia:Wembley_Stadium
"Newcastle United"	dbpedia:St_James'_Park
"Tottenham Hotspur"	dbpedia:White_Hart_Lane
"Manchester United"	dbpedia:Old_Trafford
"West Ham United"	dbpedia:Olympic_Stadium_(London)
"Chelsea"	dbpedia:Stamford_Bridge_(stadium)
"Stoke City"	dbpedia:Bet365_Stadium
"Manchester City"	dbpedia:City_of_Manchester_Stadium

Figure 19: Results for team stadiums query.

4.7 Evaluation

Our overall experience with Semantic Web tools and technologies has mixed conclusions. For one, this new way of representing our domain, allowing easily linking data to other external sources, is seen as an advantage, though it is not always clear how different

teamname	teammanager
"Queens Park Rangers"	dbpedia:Jimmy_Floyd_Hasselbaink
"Chelsea"	dbpedia:Antonio_Conte
"Stoke City"	dbpedia:Mark_Hughes
"Arsenal"	dbpedia:Arsène_Wenger
"Manchester City"	dbpedia:Pep_Guardiola
"Tottenham Hotspur"	dbpedia:Mauricio_Pochettino
"Leicester City"	dbpedia:Claudio_Ranieri
"Everton"	dbpedia:Ronald_Koeman
"West Bromwich Albion"	dbpedia:Tony_Pulis
"West Ham United"	dbpedia:Slaven_Bilić
"Newcastle United"	dbpedia:Rafael_Benítez
"Crystal Palace"	dbpedia:Alan_Pardew
"Manchester United"	dbpedia:José_Mourinho

Figure 20: Results for team managers query.

sources can be connected, for it always depends on how the authors model their domain. SPARQL is a query language somewhat similar to SQL, hence its usage didn't have a steep learning curve. However, not all tasks presented the same difficulty. Players that scored goals in seven months, for example, had challenges not only due to the functions needed, but also due to the tools not having full compliance with SPARQL 1.1. All queries involving subqueries required some extra thought, as we try to avoid them, but ultimately were considered the simplest option. Integrating with the DBpedia ontology was not as complicated as we thought and achieved good results, though we recognise the solution's lack of elegance (e.g., creating owl:sameAs properties would be more suitable). However, not all of DBpedia's teams' URIs could be automatically calculated or existed at all, which would turn this into a manual process, and thus the option was discarded.

The results for all queries were as expected. Since we are working with a data retrieval paradigm, we are able to express what we want and nothing else.

Finally, our experience with the tools available was not positive. Protégé is a single core application that relies on having all the information in memory. Queries are CPU intensive (once again, >= 80% usage was not uncommon), thus making multi-tasking hard. Moreover, it appears that every tool has their flavor of SPARQL implemented: there was no way of finding a tool that allowed a fast, user-friendly way of creating ontologies with full SPARQL 1.1 support for querying. Overall, Protégé is a good beginner tool to define ontologies, but we would not recommend it for querying them. We stuck with it as much as possible in order to facilitate the development cycle, but the likes of Apache Jena are probably a more suitable option for a wider range of querying tasks.

4.8 Semantic Web vs Information Retrieval

Semantic Web and Information Retrieval (IR) are two areas that serve distinct purposes. While the former has as objective making information spread across different sources semantically linked and, thus, understandable at machine level, the latter intends to mostly answer textual human information needs. The Semantic Web allows one to perform more complex queries that usually involve aggregation operations with data. On the other hand, for users that are exploring (i.e., have informational needs) or just do not know what they are looking for exactly, aside from some often vague textual keywords, the Semantic Web is not as appropriate, which is not helped by their more complex querying structure. For

example, if I want to find matches about some team (like the last retrieval task initially identified, about Wolverhampton mentions), a simple query in an IR system will likely give me relevant results fast. On the contrary, with an ontology, we would have to setup a SPARQL query, with all the adequate prefixes and analyze what relations we had to chain in order to query it. Another example would be querying for expressions for match reports: while IR has analyzer pipelines that facilitate textual search and the finding of similar, good enough results that could still fulfill the information need, in Semantic Web a more exact match would be required, as the system does not make any inference about what the user could want instead of what he typed, the way he typed.

On the other hand, if I want to know what players score the most goals in a specific period of the game (e.g., I'm considering an online bet), an IR system would probably not be of much help. The Semantic Web is most useful in extracting these complex statistics that are not found in common statistic web pages, which was our ultimate goal when choosing this theme and the most likely use case of our ontology by third parties.

Both areas can integrate with other existing knowledge: while this is a core value of the Semantic Web, in IR there is the possibility of adding new, specialized search engines, though this process might not be as automatic as linking data on the Web.

In the end, the Semantic Web and IR each have their own use cases and should be seen as complementary technologies that maximize the diverse users' needs.

5 CONCLUSIONS

This paper addressed three distinct milestones for a system capable of answering complex queries about historic English Premier League data: Data Preparation, Information Retrieval and Semantic Web. The collected data was cleaned, refined and its domain conceptually described to assess the main entities and associations between them. Preliminary analysis show patterns commonly associated with football matches. From this, it was possible to identify information retrieval tasks that were later answered after configuring a Solr instance with our own schema and exploring a subset of the available functionalities. Results are positive for what would be expected by a typical system user and are consistent with existing literature. With Semantic Web technologies, the domain was reanalyzed from a new perspective and an ontology that reused existing concepts from other ontologies was created, which was later queried to answer the remaining identified retrieval tasks, as well as integrated with external collections. Though results were as expected and allowed a comparison between the Semantic Web and Information Retrieval's main uses cases and advantages/limitations, the main conclusion is that Semantic Web tools often lack a standard implementation that makes them beginner-friendly and reliable for all stages of an ontology's life-cycle at the same time.

REFERENCES

[1] Apache. 2019. Apache Jena. Retrieved December 2019 from https://jena.apache.org/

[2] BBC. 2019. a simple lightweight ontology for publishing data about competitive sports events. Retrieved December 2019 from https://www.bbc.co.uk/ontologies/sport

[3] Tim Berners-Lee, James Hendler, Ora Lassila, et al. 2001. The Semantic Web. *Scientific American* 284, 5 (2001), 28–37.

[4] Jeen Broekstra. 2018. Fix GH-6 migrate to rdf4j. Retrieved December 2019 from https://github.com/protegeproject/rdf-library/pull/7

[5] Lorenz Buehmann. 2017. (SNAP SPARQL) Evaluation of HAVING solution modifier. Retrieved December 2019 from https://github.com/protegeproject/snap-sparql-query/issues/15

[6] Lorenz Buehmann. 2018. Feature request: move to RDF4J. Retrieved December 2019 from https://github.com/protegeproject/rdf-library/issues/6

[7] Cellfie. 2018. ProtÃĺgÃĺ plugin for creating OWL ontologies from spreadhseets. Retrieved December 2019 from https://github.com/protegeproject/cellfie-plugin

[8] Lucene Core. 2019. Apache Lucene Core. Retrieved November 2019 from https://lucene.apache.org/core/

[9] csnyulas. 2016. (SNAP SPARQL) Subqueries are not allowed. Retrieved December 2019 from https://github.com/protegeproject/snap-sparql-query/issues/10

[10] DAML. 2002. Soccer ontology. Retrieved December 2019 from http://www.lgi2p.ema.fr/~ranwezs/ontologies/soccerV2.0.daml

[11] Sourav Das. 2019. Top 10 Most Popular Sports in The World [Updated 2019]. Retrieved January 2020 from https://sportsshow.net/top-10-most-popular-sports-in-the-world/

[12] DBPedia. 2019. DBpedia online SPARQL endpoint. Retrieved December 2019 from https://dbpedia.org/sparql

[13] DBpedia. 2019. A shallow, cross-domain ontology. Retrieved December 2019 from https://wiki.dbpedia.org/services-resources/ontology

[14] BeautifulSoup4 documentation. 2019. Web page scraper. Retrieved October 2019 from https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[15] Solr DocValues. 2019. More efficient way of storing faceting fields. Retrieved November 2019 from https://lucene.apache.org/solr/guide/8_2/docvalues.html

[16] Steve Douglas. 2018. It is the Big 6 vs. everyone in top-heavy Premier League. Retrieved November 2019 from https://www.theglobeandmail.com/sports/soccer/article-its-the-big-6-vs-everyone-in-top-heavy-premier-league/

[17] ElasticSearch. 2019. Open Source Search and Analytics. Retrieved November 2019 from https://www.elastic.co/pt/products/elasticsearch

[18] FOAF. 2014. FOAF Vocabulary Specification. Retrieved December 2019 from http://xmlns.com/foaf/spec/

[19] Football-data. 2019. Football Betting and results archive. Retrieved October 2019 from http://football-data.co.uk

[20] The Guardian. 2019. Sports News, Comments and Results. Retrieved September 2019 from https://www.theguardian.com/uk/sport

[21] Joshua Hardwick. 2018. Google Search Operators: The complete list. Retrieved November 2019 from https://ahrefs.com/blog/google-advanced-search-operators/

[22] joew4ng. 2018. FootballNET: A football ontology. Retrieved December 2019 from https://github.com/joew4ng/footballnet

[23] Football Lineups. 2019. Football Tactics and Lineups Database. Retrieved September 2019 from https://www.football-lineups.com/

[24] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA.

[25] Apurva Nandan. 2014. A project showing the use of OWL/RDF for soccer records. Retrieved December 2019 from https://github.com/apurva3000/RDF-OWL-Soccer

[26] Newspaper3k. 2019. Article scraping and curation. Retrieved October 2019 from https://newspaper.readthedocs.io/en/latest/

[27] OntoRefine. 2019. Loading data using OntoRefine. Retrieved December 2019 from http://graphdb.ontotext.com/documentation/free/loading-data-using-ontorefine.html

[28] OpenRefine. 2019. A free, open source, powerful tool for working with messy data. Retrieved October 2019 from http://openrefine.org/

[29] MF Porter. 1980. An Algorithm for Suffix Stripping. *Program: Electronic Library and Information Systems* 14 (03 1980). https://doi.org/10.1108/eb046814

[30] Proton. [n. d.]. Proto Ontology Project. Retrieved December 2019 from http://www.ontotext.com/proton/protonext.html

[31] protÃĺgÃĺ. 2019. A free, open-source ontology editor and framework for building intelligent systems. Retrieved December 2019 from https://protege.stanford.edu/

[32] OpenRefine Reconciliation. 2019. Matching text names to database IDs. Retrieved October 2019 from https://github.com/OpenRefine/OpenRefine/wiki/Reconciliation

[33] Bertrand Rigaldies. 2018. The Solr (Multi-Terms) Synonyms Maze. Retrieved November 2019 from https://pt.slideshare.net/BertrandRigaldies/the-solr-multiterms-synonyms-maze-graphs

[34] Steve Rowe. 2017. Multi-Word Synonyms in Solr With Query-Time Support. Retrieved November 2019 from https://lucidworks.com/post/multi-word-synonyms-solr-adds-query-time-support/

[35] Schema.org. 2019. Creating, maintaining, and promoting schemas for structured data on the Internet. Retrieved December 2019 from http://schema.org/

[36] Arien Shibani. 2018. year() function DOES NOT work in SPARQL (Protege). Retrieved December 2019 from https://stackoverflow.com/questions/53474119/year-function-does-not-work-in-sparql-protege

[37] SkySports. 2019. Sports News, Transfers, Scores. Retrieved October 2019 from https://www.skysports.com/premier-league-results/2018-19
[38] Solr. 2019. Apache Solr. Retrieved November 2019 from https://lucene.apache.org/solr/
[39] Solr. 2019. The Extended DisMax Query Parser. Retrieved November 2019 from https://lucene.apache.org/solr/guide/8_2/the-extended-dismax-query-parser.html
[40] Solr. 2019. Solr Post Tool. Retrieved November 2019 from https://lucene.apache.org/solr/guide/8_2/post-tool.html
[41] Solr. 2019. Solr Schema API. Retrieved November 2019 from https://lucene.apache.org/solr/guide/8_2/schema-api.html
[42] SNAP SPARQL. 2018. An API for parsing SPARQL queries . Retrieved December 2019 from https://github.com/protegeproject/snap-sparql-query
[43] Premier League Stats. 2019. First team club statistics, team and player stats. Retrieved October 2019 from https://www.premierleague.com/stats
[44] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2008. YAGO: A Large Ontology from Wikipedia and WordNet. *Journal of Web Semantics* 6, 3 (2008), 203 – 217. https://doi.org/10.1016/j.websem.2008.06.001 World Wide Web Conference 2007Semantic Web Track.
[45] W3C. 2012. OWL 2 Web Ontology Language Manchester Syntax (Second Edition). Retrieved December 2019 from https://www.w3.org/TR/owl2-manchester-syntax/
[46] W3C. 2013. SPARQL 1.1 Overview. Retrieved December 2019 from https://www.w3.org/TR/sparql11-overview/
[47] W3C. 2014. XML Schema. Retrieved December 2019 from https://www.w3.org/2001/XMLSchema
[48] Wikidata. 2019. The free knowledge base. Retrieved October 2019 from https://www.wikidata.org/wiki/Wikidata:Main_Page
[49] Wikipedia. 2019. The free encyclopedia. Retrieved December 2019 from https://www.wikipedia.org/
[50] Wikipedia. 2019. List of Premier League Hat-tricks. Retrieved November 2019 from https://en.wikipedia.org/wiki/List_of_Premier_League_hat-tricks
[51] Zookeeper. 2019. Apache Zookeeper. Retrieved November 2019 from https://zookeeper.apache.org/

A DATASET ENTRIES EXAMPLES

Listing 14: CSV dataset example

```
Date,HomeTeam,AwayTeam,HomeFTScore,AwayFTScore
    ,HomeHTScore,AwayHTScore,Referee
16/08/2014,Arsenal,Crystal Palace,2,1,1,1,J
    Moss
16/08/2014,Leicester City,Everton,2,2,1,2,M
    Jones
16/08/2014,Manchester United,Swansea City
    ,1,2,0,1,M Dean
```

Listing 15: JSON dataset example

```
{
    "date": "12/05/2019",
    "arena": "Amex Stadium",
    "attendance": 30662,
    "home_team": {
        "name": "Brighton and Hove
            Albion",
        "lineup": [
            {"name": "Glenn Murray",
                "yellows": [], "reds
                ": [], "own_goals":
                [], "goals": [27], "
                missed_pens": [], "
                sub_off": [68]},
                ...],
        "subs": [
            {"name": "Martin Montoya
                ", "yellows": [], "
                reds": [], "own_goals
                ": [], "goals": [], "
                missed_pens": [], "
                sub_on": [84], "
                sub_off": []}, ...]
    },
    "away_team": {...},
    "report": "Highlights from
        Manchester City's win over
        Brighton in the Premier
        League. Manchester City came
        from behind to retain the
        Premier League title in style
         with a 4–1 victory at
        Brighton on the final day of
        the season. (...)"
}
```

B COLLECTION DOCUMENT EXAMPLE

Listing 16: Document example

```
{
    "date": "2015-05-24",
    "home": "Arsenal",
    "away": "West Bromwich Albion",
    "arena": "Emirates Stadium",
    "attendance": 59971,
    "home_lineup": ["David Ospina", "Hector
        Bellerin", ...],
    "away_lineup": ["Boaz Myhill", "Craig
        Dawson", ...],
    "home_subs": ["Wojciech Szczesny", "
        Laurent Koscielny", ...],
    "away_subs": ["Chris Baird", "Craig
        Gardner", ...],
```

```
    "home_scorers": ["Jack Wilshere", "Theo
        Walcott"],
    "away_scorers": ["Gareth McAuley"],
    "report": "When Jack Wilshere scored
        Arsenal's third goal after 17
        minutes (...)",
    "home_ft_score": 4,
    "away_ft_score": 1,
    "home_ht_score": 4,
    "away_ht_score": 0,
    "referee": "R Madley"}
```


C TABULAR DATASETS EXAMPLES

Listing 17: Game information example


```
season,date,home,away,arena,attendance,
    report,home_ft_score,away_ft_score,
    home_ht_score,away_ht_score,referee
2015,2015-05-24,Arsenal,West Bromwich Albion
    ,Emirates Stadium,59971,When Jack
    Wilshere scored Arsenal's third goal
    (...),4,1,4,0,R Madley
2015,2015-05-24,Aston Villa,Burnley,Villa
    Park,40792,Aston Villa supporters are
    trusting this was a case of (...)
    ,0,1,0,1,M Jones
```


Listing 18: Highligght appearance example


```
Player,Home,Away,Season,Team,Type,Minute
Hector Bellerin,Arsenal,West Bromwich Albion
    ,2015,Arsenal,Yellow_Card,48
Francis Coquelin,Arsenal,West Bromwich
    Albion,2015,Arsenal,Substitution_Off,69
Jack Wilshere,Arsenal,West Bromwich Albion
    ,2015,Arsenal,Goal,17
Jack Wilshere,Arsenal,West Bromwich Albion
    ,2015,Arsenal,Substitution_Off,77
```


Listing 19: Pure appearance example


```
Player,Home,Away,Season,Team
David Ospina,Arsenal,West Bromwich Albion
    ,2015,Arsenal
Per Mertesacker,Arsenal,West Bromwich Albion
    ,2015,Arsenal
Gabriel Paulista,Arsenal,West Bromwich
    Albion,2015,Arsenal
```