Linked Open Data Qualification for the Premier League

A study of DBpedia’s football data capabilities

Kalle Wåhlin

Bachelor’s Thesis in Computer Science
at Stockholm University, Sweden, 2021

Stockholm University

Linked Open Data Qualification for the Premier League

A study of DBpedia's football data capabilities

Kalle Wåhlin

Bachelor's Thesis in Computer Science (15 ECTS credits)
Single Subject Course
Stockholm University year 2021
Supervisor at the Department of Mathematics was Max Zeuner
Examiner was Lars Arvestad

Department of Mathematics
Stockholm University
SE-106 91 Stockholm, Sweden

Abstract

For the Web to reach its full potential as an information source its content needs to be structured for machines to interpret. This need for has pushed for a growth of Linked Open Data (LOD) and we can find today a large variation of domains represented in LOD. As for the domain of *football*, however, comprehensive datasets are difficult to find.

This study examines the football data capabilities of DBpedia – the largest cross-domain dataset in LOD – with the objective to gain insight about the usefulness of existing football data in LOD as well as challenges for the domain to grow. For this purpose we compare a sample dataset with corresponding data in DBpedia. This dataset describes six basic properties each for 619 players of the English Premier League. We analyze the ability of the DBpedia ontology to describe the properties before we extract DBpedia player data and measure its quality. Various SPARQL query techniques are used throughout the study and we implement an ad-hoc designed algorithm for identifying players in the DBpedia dataset.

While some properties (e.g. birth date) have well-defined ontology terms, others (e.g. club) are more vaguely defined, thus the DBpedia ontology fails to describe football players in a simple and consistent way. This lack of well-formed definitions makes it difficult to carry out proper data quality measurements. Despite the difficulty, the results still show that the DBpedia football data is of good quality in terms of population completeness, i.e. most players are represented in the dataset. On the downside, the property completeness is significantly low, meaning that proper descriptions are missing for many players, and thus DBpedia can not be considered a relevant source for football data.

Despite not providing useful football data itself, the DBpedia dataset, as a central hub in LOD, has the potential to be used as a stepping stone for discovering other football datasets. However, the lack of a well-developed football ontology limits the domain's expansion in LOD. A proper ontology would likely be a catalyst for the football domain to grow, both in DBpedia and in LOD in general.

Kvalificera Länkad öppen data för Premier League

En studie om DBpedias funktionalitet för fotbollsdata

Sammanfattning

För att webben ska kunna uppnå sin fulla potential som informationskälla behöver dess innehåll struktureras så att det kan tolkas av maskiner. Detta behov har gett upphov till en ökning av Länkad öppen data (LOD) och idag finns många olika domäner representerade i LOD. För domänen *fotboll* är det dock svårt att hitta omfattande dataset.

I den här studien undersöks DBpedia – det största icke domänspecifika datasetet i LOD – i egenskap av källa för fotbollsdata. Syftet är att utreda nyttan av den fotbollsdata som finns i LOD idag samt vad som krävs för att domänen ska kunna växa. För detta ändamål jämförs ett testdataset med motsvarande data i DBpedia. Datasetet beskriver sex vanliga egenskaper vardera för 619 spelare i engelska Premier League. Vi analyserar hur väl DBpedia-ontologin beskriver egenskaperna och därefter extraherar vi spelardata från DBpedia som vi sedan mäter kvaliteten på. Olika typer av SPARQL-frågor används genom studien och vi implementerar en algoritm skapad specifikt för ändamålet att identifiera spelare i DBpedias dataset.

För en del egenskaper, som födelsedatum, finns väldefinierade ontologitermer. Andra (t.ex. klubb) däremot är desto vagare definierade och därför misslyckas DBpedia-ontologin med att beskriva fotbollsspelare på ett enkelt och konsekvent sätt. Bristen på välformade definitioner försvårar datakvalitetsmätningar. Trots detta visar resultaten att DBpedias fotbollsdata håller god kvalitet vad gäller populationsfullständighet, dvs. de flesta spelarna finns representerade i datasetet. Desto sämre resultat påvisas för egenskapsfullständighet; många spelare saknar beskrivna egenskaper. DBpedia kan därför inte anses vara en relevant källa för fotbollsdata.

DBpedia är en central hubb i LOD så även om det självt saknar användbar fotbollsdata har det potential att användas som språngbräda för att upptäcka nya fotbollsdataset. Avsaknaden av en välutvecklad fotbollsontologi begränsar dock domänens utbredning i LOD. En ordentlig ontologi skulle antagligen fungera som katalysator för fotbolldomänens tillväxt, såväl hos DBpedia som LOD överlag.

Contents

1 Introduction

The *World Wide Web* (or simply *the Web*) constitutes mainly of web pages. A web page is a document that can be accessed in a web browser, either by providing a web address or – more commonly – by following a hyperlink on another web page. A hyperlink points from one web page to another and helps us find interrelated information spread across different web pages.

The Web is a gigantic source of information of any kind. However, for humans, finding and connecting all relevant pieces of information becomes a very time-consuming task. Machines on the other hand are able to quickly process large amounts of web page content. The problem is that machines lack the ability to identify and validate relevant content.

As an illustrating example, imagine we need to know all the buildings in Sweden taller than 100 meters and their architects. Unless we are lucky to find a page that has already compiled this information, we would have to find a method to browse relevant web pages, read them and store the relevant data. If we instead instruct a computer to carry out the task, the problem will not be to browse and read, but rather to identify and validate the data. This is because web pages are generally meant for humans to read and interpret.

The World Wide Web Consortium[1] (W3C) has envisioned a web where data is structured to be efficiently interpreted by machines. This web, the *Semantic Web*, can be thought of as an unlimited database that can be queried the same way we query traditional databases. (W3C 2009a)

In the Semantic Web, one single query would be enough to retrieve all the Swedish buildings taller than 100 meters and the names of their architects. It would even be possible to extract facts that are not explicitly stated, thanks to logical implications that a machine can interpret. For example, the names "Hammarbytornet" and "Hammarby radiolänktorn" both refer to the same building. Knowing that they are interchangeable, by referring to one we can access the other.

Whereas the Semantic Web is more of a vision, *Linked Data* is what embraces the vision. Linked Data is about sharing and connecting web resources[2], following a few basic principles stated by W3C. *Linked Open Data*, shortened LOD, refers to data that is shared on the web under open license and conforms to these principles. (Bizer et al. 2009)

The amount of Linked Open Data has grown rapidly in recent years and all the interlinked datasets form together a graph that is referred to as the LOD cloud. According to lod-cloud.net the cloud contained 1,239 datasets with 16,147 links in March 2019. Figure 1 shows a visual representation of the LOD cloud with the datasets categorized by domain.

[1]The inventor of the WWW, Tim Berners-Lee, founded W3C with "the mission to lead the World Wide Web to its full potential by developing protocols and guidelines that ensure the long-term growth of the Web."(W3C 2012)

[2]Web resource is a wider term than web page and refers to anything that can be retrieved from the web.(Heckmann 2006, p. 28)

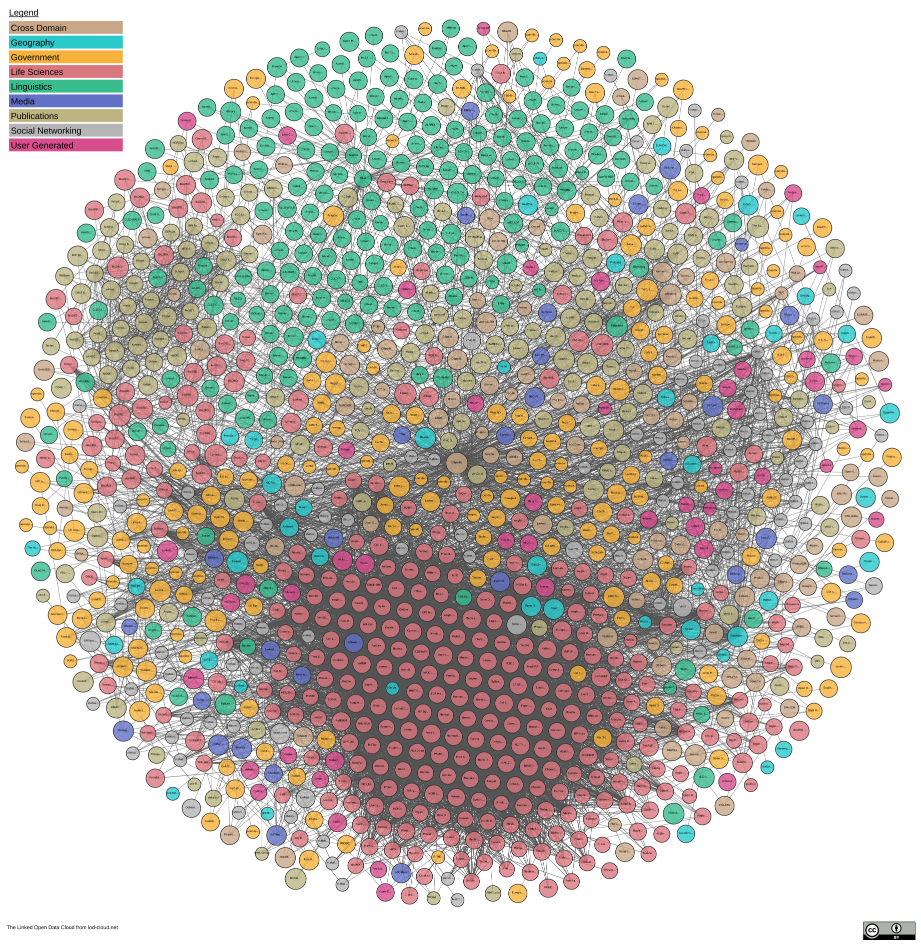


Figure 1: A visual representation of the LOD cloud provided by lod-cloud.net (McCrae 2019).

In the very middle of the cloud one can find DBpedia, a large cross-domain dataset with data extracted from Wikipedia pages. As can be seen from the image, DBpedia is linked to many other datasets, making it a central hub in the LOD cloud.

While DBpedia enables the knowledge-thirsty person to ask complex queries that span over multiple Wikipedia pages, it can also be used in software applications, like a quiz game with DBpedia-generated questions (Vega-Gorgojo 2019), or combined with Augmented Reality to display information about deceased people in cemeteries (Matuszka and Kiss 2014).

Whatever the area of use is, each application that exploits LOD requires a certain data quality level corresponding to its purpose, as is illustrated by the following example given by Zaveri et al. (2016, p. 2):

> Even datasets with quality problems might be useful for certain applications, as long as the quality is in the required range. For example, in the case of DBpedia the data quality is perfectly sufficient for enriching Web search with facts or suggestions about common

> sense information, such as entertainment topics. In such a scenario, DBpedia can be used to show related movies and personal information, when a user searches for an actor. In this case, it is rather neglectable, when in relatively few cases, a related movie or some personal fact is missing. For developing a medical application, on the other hand, the quality of DBpedia is probably insufficient.

As the LOD cloud steadily grows, problems such as inconsistency, inaccuracy, out-of-dateness and incompleteness, which considerably limit the potential of LOD, needs to be addressed. According to Rula et al. (2016, pp. 99–110), so far these issues have not been paid enough attention by the Linked Data community. To facilitate assessment of data quality in LOD, Rula et al. (2016) provide a list of various quality dimensions.

When exploring the LOD cloud at lod-cloud.net, a football enthusiast might quickly notice the domain's absence. The football domain is not mentioned much in literature on Linked Data either. Whereas other domains such as geography and music appear to have good coverage in the LOD cloud, the football domain seems under-represented and according to Bergmann et al. (2013) this applies to the sport domain in general.

Having access to football data of linked nature should be of great interest for many applications. However, with no obvious football dataset, it seems that one has to turn to cross-domain datasets for finding published linked football data.

DBpedia undoubtedly provides today linked football data, the question is whether the data is useful for applications. References that describe DBpedia's coverage of the football domain are difficult to find. Bergmann et al. (2013) describe football data in DBpedia as incomplete and unreliable, although without elaborating. Given that DBpedia has developed a lot in recent years, it is relevant to revisit the question of whether DBpedia can be used by applications as a source of linked data in the football domain.

If DBpedia data is to be used by an application, it should be assessed first. The quality of the data needs to be satisfying for the task at hand. Whether it's about generating football quiz questions, enhancing the experience at a football stadium with augmented information about the teams and players, or answering sophisticated, complex questions.

This study aims to assess the maturity of the football domain in Linked Open Data. For this purpose, we will look into DBpedia's football data capabilities. We will analyze the DBpedia ontology and evaluate fundamental and well-known quality aspects of a given sample dataset.

From the findings of the study conclusions will be drawn about the maturity of the domain in DBpedia; how well-developed it is in terms of completeness and correctness, and its relevance as a source of information for other applications.

The conclusions will hopefully provide guidance for further development of the football domain in Linked Open Data.

2 Theory

This section presents the theoretic background for concepts occurring in the study. First, Linked Data and its building blocks are explained. Subsequently follows a description of DBpedia and finally different aspects of linked data quality are explained.

2.1 Linked Data

As mentioned in the introduction, Linked Data is an implementation of the ideas of a Semantic Web. The four Linked Data principles, as stated by W3C founder Berners-Lee (2006), are:

- Use URIs as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
- Include links to other URIs so that they can discover more things.

We will explain these four points in sections 2.1.1–2.1.4.

2.1.1 URI, namespace and prefix

A URI, short for Uniform Resource Identifier, refers to a resource of any kind. The resource can be something physical such as a building or digital such as a document or an image. It can even be an abstract concept such as "king" or "red".

Usually, URIs coincide with URLs. A URL (Unique Resource Locator) is used for locating a web page, i.e. it is the *web address* of that page. For example, `https://www.example.com/Lionel_Messi` could be the URL for finding a document about the football player Lionel Messi on the web. However, the very same string could also be a URI referring to the real world football player that is Lionel Messi.

By using a HTTP URI, the resource gets a web representation, and it becomes accessible for anyone. This is what the second principle is about. Many different resources are often represented within the same *namespace*. Namespaces allows for reuse of words in URIs by putting them in different contexts. For example, the word "king" might be used within the namespace `https://titles.com/` to form the URI `https://www.titles.com/king` which refers to the resource that is the title of a male monarch. Used in another namespace, e.g. `https://www.companies.com/king` it might refer to the company named King.

In Linked Data, prefixes are often used in order to avoid writing long HTTP URIs. A prefix can replace a namespace so that a URI can more easily be referenced. If the prefix `ex` is defined to represent the namespace `https://www.example.com/`, then the URI `https://www.example.com/Lionel_Messi` can be referenced by just writing `ex:Lionel_Messi`.

OpenLink Software (2020) provide a list of prefixes and the full namespaces that they represent. The list can be used by the reader as a dictionary to look up the prefixes appearing in this study.

2.1.2 The RDF data model

The Resource Description Framework (RDF) data model is fundamental to Linked Data. It allows for resources to be described and linked together in a simple and uniform structure.

An RDF-statement is a subject-predicate-object triple that describes a property of the subject. The subject of a triple is always a URI, e.g. `ex:Lionel_Messi`. The predicate is also a URI and has the role of expressing in what way the object is related to the subject, i.e. what property is described. The object is the property value and can be either another URI or a plain literal. For example, if the predicate says that the object is the nationality of the subject, then the object is probably a URI referencing that country. However, if the height of the subject is expressed, the object is more likely a literal such as "1.70".

While there are several valid formats for storing RDF-data, the subject-predicate-object triple model always applies. RDF-triples form a directed labeled graph. Each triple is a pair of vertices with an edge between them. The edge is labeled with the predicate and points from the subject to the object.

Figure 2 shows an example graph formed by a few triples. The URI `ex:Lionel_Messi` has been made up, however the other exist. The predicates as well as the *class* `dbo:SoccerPlayer` are acknowledged *ontology* terms. In the example, namespaces have been replaced by prefixes.

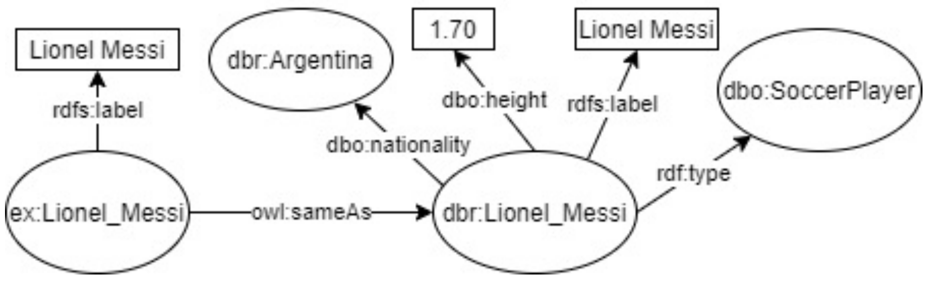


Figure 2: Example RDF-graph. Ovals and arrows are URIs and rectangles are literals.

2.1.3 Ontologies

Ontologies provide the terms for describing resources in a linked dataset in a uniform and accurate way. Meta-descriptions in ontologies define the meaning of the terms that are used in RDF-statements, i.e. *properties of the properties*.

Different ontologies can be developed for different domains of discourse, however the core of any ontology conforms to standards provided by W3C. These standards are RDF Schema (RDFS) and Web Ontology Language (OWL). They are themselves sets of RDF-statements and provide means of generically describing properties, relations and classes in a concise and logical manner.

Looking at the example graph in Figure 2 we have for example the predicate `rdfs:label` from RDFS. This term is the universal standard for expressing the name of a resource. The meta-description of `rdfs:label` states that it has the `rdf:range` of `rdfs:Literal`, meaning that the object of a triple with `rdfs:label` as predicate is expected to be a literal and not a URI. Both `rdf:range` and `rdfs:Literal` in turn are also terms of RDFS and have their own meta-descriptions. For further explanation of RDFS see Brickley and Guha (2014).

Another term from RDFS that we find in the example is `rdf:type`. This predicate states that the subject is a member of a *class* (the object). The class in the example, `dbo:SoccerPlayer` is from the DBpedia ontology. We say that the resource in the subject *is of type* or *is a* `dbo:SoccerPlayer`.

The predicate `dbo:height` is also from the DBpedia ontology. It has a *typed literal* range, namely `xsd:double`. As opposed to when the range is `rdfs:Literal`, a typed literal is expected to be of a certain datatype. In this case a double, i.e. the property `dbo:height` is expressed in meters. The datatypes are identified by URIs in the XML schema namespace (`xsd:`) and other examples are `xsd:date`, `xsd:int` and `xsd:string`.

The `owl:sameAs` term is used for saying that two resources are identical, i.e. the URIs reference the same entity. This is useful for linking together resources from different namespaces having the same meaning. For a full description of OWL see Dean and Schreiber (2004).

Other examples of well-known ontologies that are not domain-specific are Dublin Core and Simple Knowledge Organization System (SKOS). They are described in DCMI Usage Board (2020) and Miles and Bechhofer (2009) respectively.

A term often used interchangeably with ontology is *vocabulary* (W3C 2009b), although the former tends include more complex structures (DuCharme 2011, p. 39). For the sake of simplicity, in this paper ontology refers to both.

2.1.4 SPARQL

SPARQL is a declarative language for querying RDF-data. Collections of RDF-data can be loaded into SPARQL *endpoints*, which are web services that allow the data to be queried (DuCharme 2011, p. 14). The key when constructing SPARQL queries is to specify triple patterns. Triples conforming to those patterns are retrieved and the `SELECT` keyword lets one choose which components of the triples to return.

Two example queries are shown in Figure 3. In 3a, we simply state that we want the subject component of triples having `rdfs:label` as predicate and “Lionel Messi” as object. The triple patterns are always enclosed in braces. The query in 3b is constituted by several triple patterns. The first line states that we want URIs of type `dbo:SoccerPlayer`. Those resources are *bound* to the variable ?Uri. The “a” in this first pattern is syntactic sugar for `rdf:type`. Furthermore, we require that the resources bound to ?Uri are also the subject of triples with `dbo:nationality` as predicate, i.e. that they are linked to a country. The third line states that a name must also be associated with resources bound to ?Uri. These names are bound to the variable ?Name. The last line declares that the country associated with the URIs must have the name Argentina. Finally, with `SELECT` the query is instructed to return pairs of URIs and literals bound to ?Uri and ?Name respectively, i.e. Argentinian football players and their names.

```
SELECT ?Uri
{?Uri rdfs:label "Lionel Messi"}
```

(a) A simple query that returns resources named "Lionel Messi".

```
SELECT ?Uri ?Name
{
   ?Uri a dbo:SoccerPlayer ;
      dbo:nationality ?Country ;
      rdfs:label ?Name .
   ?Country rdfs:label "Argentina"
}
```

(b) A query returning Argentinian football players and their names.

Figure 3: SPARQL query examples.

A useful function in SPARQL is `VALUES`. This keyword makes it possible to decide beforehand which values should be bound to a variable. An example query with `VALUES` is shown in Figure 4. We bind here "Lionel Messi", "Kylian Mbappé" and "Mohamed Salah" to the variable ?Name, thus saying that we want URIs having either of those names.

```
SELECT ?Uri
{
   VALUES ?Name {"Lionel Messi" "Kylian Mbappé" "Mohamed Salah"}
   ?Uri rdfs:label ?Name
}
```

Figure 4: Example query using `VALUES`.

Other SPARQL functions frequently used throughout this study are e.g. `COUNT`, `FILTER`, `UNION` and `OPTIONAL`. Explanations of those and many other SPARQL functions can be found in Seaborne and Harris (2013). DuCharme (2011) is also a recommended reading for learning more about SPARQL and RDF-data in general.

SPARQLWrapper is a SPARQL endpoint interface to Python. It allows for custom queries to be constructed at run-time and processing of the returned result sets. More about SPARQLWrapper can be found in *SPARQLWrapper Documentation* (2020).

2.2 DBpedia

DBpedia is an open community project started in 2007. The main contributors are the University of Leipzig, Mannheim University and some privately owned companies like OpenLink Software and Semantic Web Company. There are other actors that also contribute to the project via the DBpedia association. (DBpedia 2009)

DBpedia's objective is to extract information from Wikipedia pages and convert it to RDF-triples. Each Wikipedia page is mapped to a URI in the `http://dbpedia.org/resource/` (`dbr:`) namespace, e.g. `https://en.wikipedia.org/wiki/Lionel_Messi` is mapped to the resource with URI `dbr:Lionel_Messi`. These URIs constitute the subjects of RDF-triples while appropriate predicates and objects correspond to information in the Wikipedia page, mostly from the infoboxes. DBpedia is multilingual and has its own ontology based on

OWL. Information extracted from Wikipedia pages is mapped to terms in this ontology.

New DBpedia datasets are released continually and can be downloaded. They are also loaded into the public endpoint at `http://dbpedia.org/sparql` and can be queried directly from there. However, the dataset currently loaded into the public endpoint is from 2016 and thus not up-to-date (DBpedia 2018).

DBpedia Live is an extension of DBpedia that better reflects the current state of Wikipedia by updating the dataset accordingly whenever a Wikipedia article is edited. This dataset can be queried from the live endpoint at `http://live.dbpedia.org/sparql`. Resources in DBpedia Live are also contained in the `http://dbpedia.org/resource/` namespace, meaning that URIs are the same in both DBpedia and DBpedia Live. As opposed to DBpedia, DBpedia Live supports English language only (i.e. it contains only data from the English Wikipedia).

When querying a DBpedia endpoint it is possible to specify the *default graph*. Different graphs are different subsets of the dataset loaded into the endpoint. Specifying the relevant graph can speed up the queries as it narrows down the search space. For example, the DBpedia ontology can be queried from the graph `dbr:classes#`. If using the DBpedia Live endpoint it can be a good idea to set the default graph to `http://live.dbpedia.org/` instead of `http://dbpedia.org/` as the latter contains a lot of extra data from other datasets.

DBpedia endpoints are powered by Virtuoso from OpenLink Software. A special feature with Virtuoso-powered endpoints is the `bif:contains` keyword which allows for free-text searching in SPARQL queries. Similar generic SPARQL functions exist (`REGEX`, `CONTAINS`), however `bif:contains` is more efficient. (DuCharme 2011, p. 225)

2.3 Linked data quality aspects

Unlike relational database schemas, ontologies do not have the power to prevent incorrect or inconsistent data from being added to an RDF dataset. As a consequence, a posteriori qualification of the data is needed. To this end, Rula et al. (2016) have compiled a list of the most common linked data quality aspects along with metrics for quantitatively evaluating them. The aspects evaluated in this study are:

- Population completeness
- Property completeness
- Syntactic accuracy
- Semantic accuracy

Population completeness refers to the extent to which real-world entities of a particular type is present in the dataset. The suggested metric is the ratio between the amount of resources in the dataset and the amount of real-world entities. This requires a complete reference dataset for comparison and implicitly also identification of the corresponding resources.

Property completeness is measured for properties associated with resources of a certain type. The suggested metric is, for a given property, the ratio between the amount of resources having this property represented and the total amount of resources. To enable property completeness measurements, the resources of interest, as well as predicates accurately expressing their properties, needs to be identified.

Syntactic accuracy refers to the extent to which literals conform to given syntactical rules. In Linked Data, this usually means whether or not literals are compatible with predicates having a typed literal range. One suggested metric is the ratio between the amount of values associated with a given predicate having correct data type and the total amount of values associated with that predicate.

Semantic accuracy refers to the extent to which RDF-triples reflect real-world facts. A reliable reference dataset is needed for validation and the predicate most accurately expressing a given property needs to be identified. Rula et al. (2016) do not provide a preferred metric for semantic accuracy. In this study we measure this aspect as the ratio between the amount of semantically correct values and the total amount of values for a given property.

3 Querying football data in DBpedia

There are several prerequisites for conducting rigid data quality measurements. Not least, a substantial set of queries had to be constructed and run against DBpedia. The list below briefly sketches the process in chronological order.

1. A reliable reference dataset with football player data was collected to be used for comparison with corresponding DBpedia data.

2. DBpedia Live was explored for ontology terms that describe the sample of football players.

3. Using the data retrieved in step 1, and the knowledge gained in step 2, an algorithm for identifying football players as DBpedia resources was implemented. The algorithm yields a measure for population completeness.

4. Properties of the resources identified in step 3 were assessed with respect to syntactic and semantic accuracy and completeness.

Each step is described in more detail in section 3.1–3.4.

3.1 The reference data

A dataset consisting of 622 players of the English Premier League (EPL) was collected from the league's official website premierleague.com. The website provides information pages for each player from where data could be extracted. A Python script was written to perform this task. Six properties of such type that are typically found in short summaries/infoboxes were collected for each player.[3] The Python library Beautiful Soup was used for identifying the property values in the HTML-code of each player page. A sample of the retrieved player data is shown in Table 1.

Table 1: Each row represent a player and each column a property.

Club	**Number**	**Name**	**Birthdate**	**Position**	**Nationality**
Arsenal	1	Bernd Leno	04/03/1992	Goalkeeper	Germany
Liverpool	4	Virgil van Dijk	08/07/1991	Defender	Netherlands
Manchester City	17	Kevin De Bruyne	28/06/1991	Midfielder	Belgium
Manchester United	10	Marcus Rashford	31/10/1997	Forward	England

3.2 Exploring the DBpedia ontology

To be able to identify the collected set of EPL players in DBpedia it was first necessary to investigate how their properties would typically be represented, i.e. how the resources are linked to their properties and what predicates express these links. This was done querying the DBpedia Live graph (*DBpL*) first for relevant classes, then for commonly used predicates that express properties of members of those classes. Meta-descriptions of the ontology terms were queried

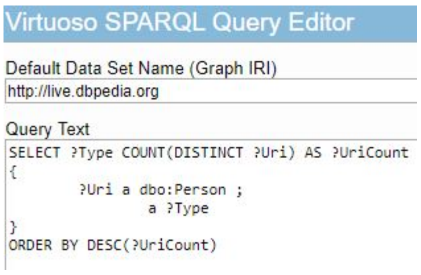
[3] A few players did not have their number or birth date represented at EPL's website. In those cases, the data was retrieved manually from transfermarkt.com.

from the DBpedia ontology graph. The queries were run directly in the web interface at `http://live.dbpedia.org/sparql`.

Understanding how the players are represented in DBpedia required much work constructing relevant queries. A variety of query techniques were used for finding, comparing and verifying the usage of different ontology terms. A selection of the queries along with partial or full results are presented in Figures 5–8. Since $DBpL$ is a dynamic dataset, all query results are temporary, thus we refer to them as something that *was* rather than *is*.

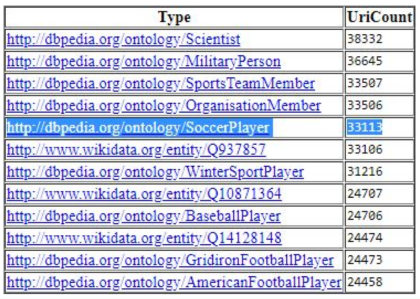
3.2.1 Finding football player classes

The first task was to identify resources being football players. We define L as the set of all resources in $DBpL$ and we let S_c denote the subset of resources S being of type c. Querying for the most common resource types overall in $DBpL$ revealed the class `dbo:Person` with ca 500000 members. Equivalent classes from other ontologies also appeared in the query result, however with E representing the set of resources belonging to any of those classes, $|E \setminus L_{dbo:Person}|$ was negligible. Thus only `dbo:Person` was used in the next query, which again ranked the most common types, but this time only for $L_{dbo:Person}$. The generated query result revealed the class `dbo:SoccerPlayer` with ca 33000 members. We refer to this class as SP onwards.


Virtuoso SPARQL Query Editor
Default Data Set Name (Graph IRI)
http://live.dbpedia.org
Query Text

```
SELECT ?Type COUNT(DISTINCT ?Uri) AS ?UriCount
{
        ?Uri a dbo:Person ;
                a ?Type
}
ORDER BY DESC(?UriCount)
```

(a) The query counts members of different classes.



Type	UriCount
http://dbpedia.org/ontology/Scientist	38332
http://dbpedia.org/ontology/MilitaryPerson	36645
http://dbpedia.org/ontology/SportsTeamMember	33507
http://dbpedia.org/ontology/OrganisationMember	33506
http://dbpedia.org/ontology/SoccerPlayer	33113
http://www.wikidata.org/entity/Q937857	33106
http://dbpedia.org/ontology/WinterSportPlayer	31216
http://www.wikidata.org/entity/Q10871364	24707
http://dbpedia.org/ontology/BaseballPlayer	24706
http://www.wikidata.org/entity/Q14128148	24474
http://dbpedia.org/ontology/GridironFootballPlayer	24473
http://dbpedia.org/ontology/AmericanFootballPlayer	24458

(b) Partial query result.

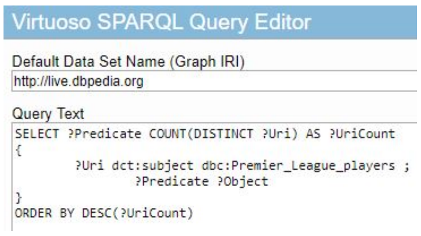
Figure 5: Ranking of common classes associated with members of `dbo:Person`*.*

Ranking the most common types associated with L_{SP} yielded no interesting new classes. Querying the most common predicates among L_{SP} on the other hand showed that most of these resources were provided with the `dct:subject` predicate, which linked them to various SKOS concepts (classes). The class `dbc:English_Football_League_players` was discovered among the more common classes and this in turn led to the detection of `dbc:Premier_League_players`, another SKOS concept with 4721 members. We call this class PP.

3.2.2 Finding property representations

At this stage, having found one class of football players in general and another of EPL players in particular, the next task was to find descriptions of members of those classes. A query showed that $L_{PP} \nsubseteq L_{SP}$, meaning that conclusions about one class would not necessarily apply to the other. Therefore, both classes had to be investigated further in search of property descriptions for their members.

The properties were found by ranking the most used predicates among L_{PP} and L_{SP} respectively. We call the set of players in either class $H = L_{PP} \cup L_{SP}$ and we let S^{ρ} denote the subset of resources S having the property ρ. The majority of properties were directly linked to a player, while some were linked via intermediate URIs.



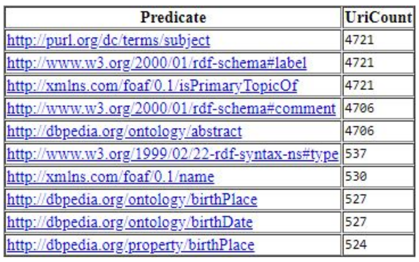
Virtuoso SPARQL Query Editor

Default Data Set Name (Graph IRI)
http://live.dbpedia.org

Query Text

```
SELECT ?Predicate COUNT(DISTINCT ?Uri) AS ?UriCount
{
        ?Uri dct:subject dbc:Premier_League_players ;
             ?Predicate ?Object
}
ORDER BY DESC(?UriCount)
```

(a) The query counts how many URIs have each predicate.



Predicate	UriCount
http://purl.org/dc/terms/subject	4721
http://www.w3.org/2000/01/rdf-schema#label	4721
http://xmlns.com/foaf/0.1/isPrimaryTopicOf	4721
http://www.w3.org/2000/01/rdf-schema#comment	4706
http://dbpedia.org/ontology/abstract	4706
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	537
http://xmlns.com/foaf/0.1/name	530
http://dbpedia.org/ontology/birthPlace	527
http://dbpedia.org/ontology/birthDate	527
http://dbpedia.org/property/birthPlace	524

(b) Partial query result.

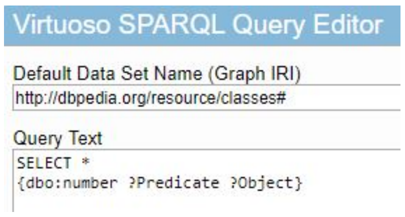
Figure 6: Ranking of common predicates for members of `dbc:Premier_League_players`.

The predicate `dbo:birthDate` links a `dbo:Person` (the domain) to its birth date, which is expected to be a literal of data type `xsd:date` (the range). A predicate with a similar URI, `dbp:birthDate`, appeared to be an alternative. However the ontology contained no meta-description for this predicate. Furthermore, the amount of resources $|H^{dbp:birthDate} \setminus H^{dbo:birthDate}|$ was insignificant. Thus `dbp:birthDate` was considered superfluous for describing birth dates of the players.

Position was also linked directly to player resources, either via `dbo:position` or `dct:subject`. Neither predicate had a defined domain/range, however URIs representing football positions could be found in the object position of triples with these two. We let $\rho(S)$ denote the value(s) associated with property ρ for a resource, or a set of resources, S. A player resource $h \in H$ could for example have `dbo:position`(h) = `dbr:Forward_(association_football)` or `dct:subject`(h) = `dbc:Association_football_midfielders`. Many players being linked to its position via `dbo:position` were not linked via `dct:subject` and vice versa, so both predicates were in this case considered useful.

The two most common ways of linking a player to its name was with `rdfs:label` or `foaf:name`. However the former predicate had better representation among player resources and should be the preferred.

The predicate `dbo:number` links a `dbo:Athlete` (superclass of `dbo:SoccerPlayer`) to its number, which is expected to be a literal of data type `xsd:string`.



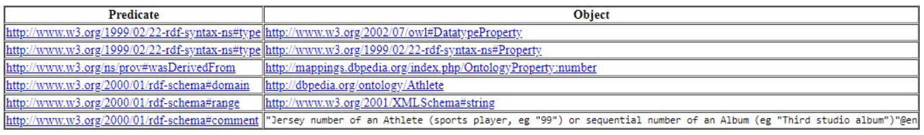
Default Data Set Name (Graph IRI)

http://dbpedia.org/resource/classes#

Query Text

```
SELECT *
{dbo:number ?Predicate ?Object}
```

(a) Query for every predicate-object pair associated with `dbo:number`*.*



Predicate	Object
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#DatatypeProperty
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/1999/02/22-rdf-syntax-ns#Property
http://www.w3.org/ns/prov#wasDerivedFrom	http://mappings.dbpedia.org/index.php/OntologyProperty:number
http://www.w3.org/2000/01/rdf-schema#domain	http://dbpedia.org/ontology/Athlete
http://www.w3.org/2000/01/rdf-schema#range	http://www.w3.org/2001/XMLSchema#string
http://www.w3.org/2000/01/rdf-schema#comment	"Jersey number of an Athlete (sports player, eg "99") or sequential number of an Album (eg "Third studio album")"@en

(b) Partial query result.

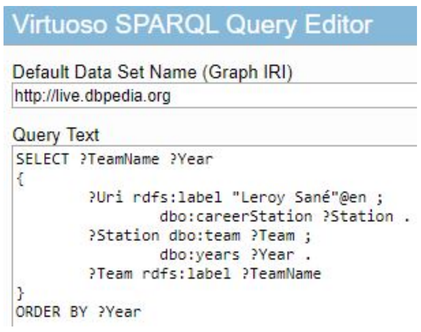
Figure 7: Information about the ontology term `dbo:number`*.*

A predicate for explicitly expressing nationality could not be found among the most used for player resources. Instead, the predicate `dbp:nationalteam` was considered. Although not being defined in the DBpedia ontology (i.e. it lacked meta-descriptions), it was found to link players to different national football teams. Thus it could be useful to find the nationality of players that have represented their national team at some point. Most of the values `dbp:nationalteam`(H) were URIs with country names contained in their labels, e.g. ”France national football team”.

The predicate `dbo:birthPlace` was considered the best alternative, should the player not be associated with a national team. It is used for linking a `dbo:Person` to a `dbo:Place`, the place where the person was born. Among `dbo:birthPlace`(H) some were URIs representing countries, others were URIs representing places within a country such as cities. In the latter case, the place was usually linked to the country where it is located with the predicate `dbo:country`. A player's birth country was thus either directly linked to the player resource or via an intermediate URI. Birth country should in many cases correspond to nationality, so `dbo:birthPlace` was reckoned a somewhat accurate way of accessing the nationality of a player.

There were several alternatives for representing the club of a player. Two predicates, `dbo:clubs` and `dbo:team`, were dismissed because they seemed to link to any club the player has represented and not just the current (both linked a player to ∼6 different clubs on average). These links were *direct* and thus not revealing any additional information about the relation between the player and club, such as time span, that can be used for deciding which club is the current. Despite not being defined in the ontology, the self-explanatory `dbp:currentclub` was reckoned more accurate. This predicate linked player resources to only ∼1.1 different clubs on average.

Another predicate that was considered useful for finding the current club of a player was `dbo:careerStation`. It links a `dbo:Person` to a `dbo:CareerStation`, which holds information about a step in the player's career. The ontology did not reveal how that information is expected to be provided, however the most common predicates associated with career stations were found to be `dbo:team` and `dbo:year`. By comparing the team and year of a player resource's career stations with information displayed in the player's Wikipedia page, it was concluded that it corresponds to the year when the player first represented the team. Hence, a player's current club could be retrieved by querying for the club of the career station having the most recent year associated with it. For career stations CS = `dbo:careerStation`(H) the associated teams `dbo:team`(CS) turned out to be a mix of clubs and national teams, meaning that in some cases also nationality could be found via `dbo:careerStation`.



Virtuoso SPARQL Query Editor

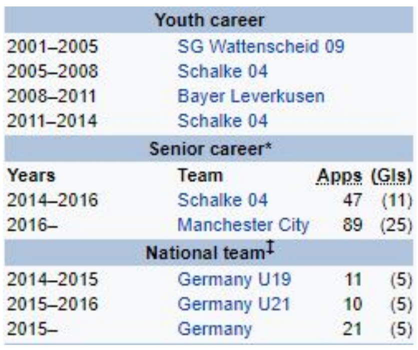
Default Data Set Name (Graph IRI)

http://live.dbpedia.org

Query Text

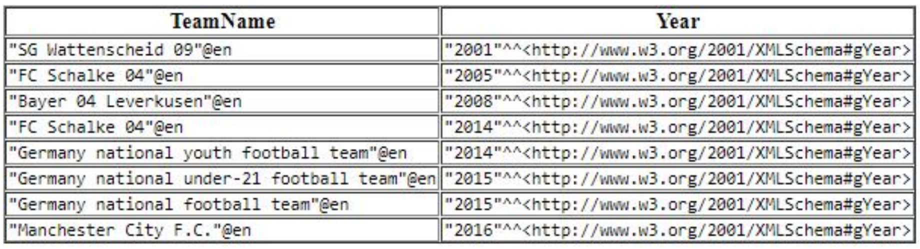
```
SELECT ?TeamName ?Year
{
        ?Uri rdfs:label "Leroy Sané"@en ;
             dbo:careerStation ?Station .
        ?Station dbo:team ?Team ;
             dbo:years ?Year .
        ?Team rdfs:label ?TeamName
}
ORDER BY ?Year
```

(a) Query every team-year pair of career stations associated with resources named Leroy Sané.



Youth career		
2001–2005	SG Wattenscheid 09	
2005–2008	Schalke 04	
2008–2011	Bayer Leverkusen	
2011–2014	Schalke 04	
Senior career*		
Years	**Team**	**Apps (Gls)**
2014–2016	Schalke 04	47 (11)
2016–	Manchester City	89 (25)
National team‡		
2014–2015	Germany U19	11 (5)
2015–2016	Germany U21	10 (5)
2015–	Germany	21 (5)

(b) The career of Leroy Sané as displayed in Wikipedia.



TeamName	Year
"SG Wattenscheid 09"@en	"2001"^^<http://www.w3.org/2001/XMLSchema#gYear>
"FC Schalke 04"@en	"2005"^^<http://www.w3.org/2001/XMLSchema#gYear>
"Bayer 04 Leverkusen"@en	"2008"^^<http://www.w3.org/2001/XMLSchema#gYear>
"FC Schalke 04"@en	"2014"^^<http://www.w3.org/2001/XMLSchema#gYear>
"Germany national youth football team"@en	"2014"^^<http://www.w3.org/2001/XMLSchema#gYear>
"Germany national under-21 football team"@en	"2015"^^<http://www.w3.org/2001/XMLSchema#gYear>
"Germany national football team"@en	"2015"^^<http://www.w3.org/2001/XMLSchema#gYear>
"Manchester City F.C."@en	"2016"^^<http://www.w3.org/2001/XMLSchema#gYear>

(c) The query result. Each row is associated with a `dbo:CareerStation` of `dbr:Leroy_Sané`.

Figure 8: The career stations in DBpedia correspond well to the information in Wikipedia's infobox.

A significantly smaller proportion of L_{PP} had appropriate predicates to describe their attributes than L_{SP}. However, `dbo:abstract` was well represented among L_{PP}. This predicate links a resource to a `rdfs:langString` that is an abstract about the resource in question. A pattern was recognized in abstracts of L_{PP}: birth date and current club were contained in the opening sentences. The `dbo:abstract` predicate was therefore considered useful in the cases where other predicates to describe those properties are missing.

To sum up the findings presented in this subsection, Figure 9 shows an extensive graph of how a player of the reference dataset could typically be represented with RDF-triples in DBpedia. The graph implements the syntax suggested by Addlesee (2018).

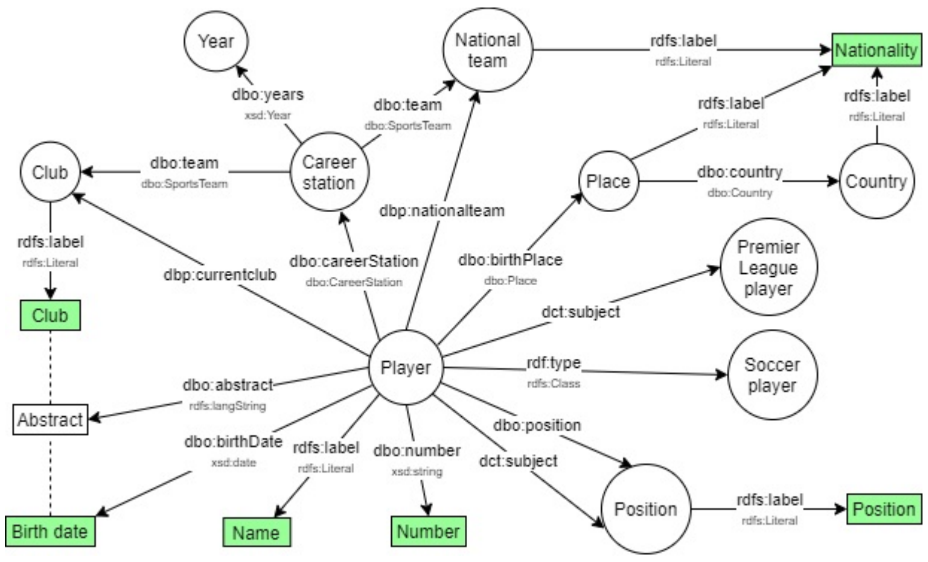


Figure 9: A visual RDF-graph of a hypothetical EPL player in DBpedia. Circles are resources, rectangles are literals and arrows are predicates. The range of a predicate, if defined, is shown in small font under a predicate URI. The dashed lines from the Abstract literal indicate that it contains birth date and club, however there is no RDF-link to those literals.

3.3 Implementing an algorithm for identifying DBpedia resources

From the findings presented in section 3.2, relevant queries could be constructed and used as the core component of an algorithm that matches players with their DBpedia resources. The idea was to, for each reference player, first query for DBpedia resources that match *one* of the six properties and then see which of those resources best matches the remaining five.

Name was reckoned the best property to start matching resources by. Firstly, it has the advantage of narrowing down the set of possible candidates to a reasonable amount. Secondly, `rdfs:label` was well-represented among player resources compared to other relevant predicates. Matching on e.g. *Birthdate* in the first place would risk filtering out the best candidate, since resources lacking `dbo:birthDate` would not be found.

As new football players emerge continually, the candidates were queried from *DBpL*. However, complementing data about the candidates was queried from the non-live endpoint (we refer to this static dataset as *DBp*), since during the implementation it was discovered that player resource descriptions are often richer in *DBp* than in *DBpL*.

The algorithm was written in Python and make use of SPARQLWrapper for embedded SPARQL querying. A module was implemented to enable automatic reading and modification of template query files before running them against DBpedia. This module also include functions for handling different types of query result sets. Several other modules were also implemented for further processing of the queried data.

We define R as the set of EPL players in the reference dataset and we refer to a player $r \in R$ by its name. For each reference player $r \in R$, the algorithm does the following:

1. Queries $DBpL$ for resources that match r.
2. Queries DBp for complementing data about the matched resources.
3. Merges queried data from $DBpL$ and DBp.
4. Compares descriptions of each matched resource with the description of r in R.
5. Chooses the best match among the candidate resources.

3.3.1 Finding possible matches in DBpedia Live

In the first step of the algorithm, the player r is inserted to a template query that retrieves all resources matching r by name. Instead of doing exact string matching, the query uses `bif:contains` in order to include labels containing more than just the name, e.g. "John Smith (footballer, born 1971)". The label has to contain both first and last name as searching for labels containing either would be too wide. However, if there are several last names only one of them needs to be contained. The query was designed so that "John Adams Smith" matches both "John Smith" and "John Adams" but not "George Smith" or "John Phillips". In this way different variations of a name could be matched, while the search space was being kept reasonably restricted.

We let C represent the set of candidates (URIs) for being the DBpedia resource representing r. For each $c \in C$, the query continues to search for remaining properties. As described in section 3.2 and illustrated in Figure 9, the properties can be retrieved in various ways. The query was designed to examine any possible route to a property and include all values that it finds in the result set. Thus, several values can be associated with one property. Even properties retrievable only in one way can have several associated values, e.g. if there is one triple where `dbo:number`(c) = "9" and another where `dbo:number`(c) = "10". Values associated with the same property are grouped together in the result set. Along with values found for each property, the query result includes `dbo:abstract`(c) and indicators if c is a `dbo:SoccerPlayer` or `dbc:Premier_League_players` (with "Yes" or empty string). The query in its entirety can be viewed in Figure 10.

```
SELECT DISTINCT ?Uri ?Label ?premPlayer ?soccerPlayer
GROUP_CONCAT(DISTINCT xsd:date(?date);separator='|') AS ?Birthdate
GROUP_CONCAT(DISTINCT ?no;separator='|') AS ?Number
GROUP_CONCAT(DISTINCT REPLACE(?countryLabel,
   " national.*", "", "i");separator='|') AS ?Nationality
GROUP_CONCAT(DISTINCT ?posLabel;separator='|') AS ?Position
GROUP_CONCAT(DISTINCT ?clubLabel;separator='|') AS ?Club
GROUP_CONCAT(DISTINCT ?abs;separator='|') AS ?Abstract
{
   ?Uri rdfs:label ?Label .
   ?Label bif:contains "$1$"
   BIND(IF(EXISTS{?Uri dct:subject dbc:Premier_League_players},
      "Yes", "") AS ?premPlayer)
   BIND(IF(EXISTS{?Uri a dbo:SoccerPlayer}, "Yes", "") AS ?soccerPlayer)
   OPTIONAL{?Uri dbo:birthDate ?date}
   OPTIONAL{?Uri dbo:number ?no}
   OPTIONAL
   {
      {
         ?Uri dbo:birthPlace ?country .
         ?country rdfs:label ?countryLabel
      }
      UNION
      {
         ?Uri dbo:birthPlace ?place .
         ?place dbo:country ?country .
         ?country rdfs:label ?countryLabel
      }
      UNION
      {
         ?Uri dbp:nationalteam ?nteam .
         ?nteam rdfs:label ?countryLabel
      }
      UNION
      {
         ?Uri dbo:careerStation ?cs .
         ?cs dbo:team ?team .
         ?team rdfs:label ?countryLabel .
         ?countryLabel bif:contains "national"
      }
   }
   OPTIONAL
   {
      {
         ?Uri dbo:position ?pos .
         ?pos rdfs:label ?posLabel
      }
      UNION
      {
         ?Uri dct:subject ?pos .
         ?pos skos:broader dbc:Association_football_players_by_position ;
            rdfs:label ?posLabel
      }
   }
   OPTIONAL
   {
      {
         SELECT ?Uri ?clubLabel
         {
            ?Uri dbo:careerStation ?cs1 ;
               dbo:careerStation ?cs2 .
            ?cs1 dbo:team ?club ;
               dbo:years ?year .
            ?club rdfs:label ?clubLabel .
            FILTER(!REGEX(?clubLabel, "national", "i"))
            ?cs2 dbo:team ?team ;
               dbo:years ?year2 .
            ?team rdfs:label ?teamLabel
            FILTER(!REGEX(?teamLabel, "national", "i"))
         }
         GROUP BY ?Uri ?clubLabel ?year
         HAVING (?year = MAX(?year2))
      }
      UNION
      {
         ?Uri dbp:currentclub ?club .
         ?club rdfs:label ?clubLabel .
      }
   }
   OPTIONAL{?Uri dbo:abstract ?abs}
}
```

Figure 10: The query first finds URIs that match the inserted name. Then it looks for various properties of those URIs. 1 is the placeholder for where the player name is inserted.

3.3.2 Complementing with data from non-live DBpedia

Descriptions of player resources varied depending on whether $DBpL$ or DBp was queried. A resource missing relevant predicates in $DBpL$ often had them present in DBp instead. We define D the set of resources in $DBpL$. For example, 100% of D_{PP} are provided with `dbo:birthDate` as compared to ca 11% of L_{PP}.

Accordingly, DBp is queried for complementing data about the candidates C found in $DBpL$. Due to DBp being outdated, only the over time more consistent properties are queried for. These include $Birthdate$, $Nationality$, $Position$ and being a football player (i.e. a `dbo:SoccerPlayer`). The triple patterns are the same as in the first query (Fig. 10), however with the URIs now being known, no name matching is needed. The candidates are inserted together with the `VALUES` keyword and the query looks up their properties.

In some cases $|C|$ was too large to be handled by one query (e.g. the player name "Fred" matches ca 6000 resources). Therefore, a maximum of 1000 URIs are queried at a time. Since DBp contains different languages, filters were also applied on string literals (with the `FILTER` keyword) to retrieve only those that are in English. Table 2 shows different descriptions of the same URI. This clearly demonstrates how DBp can be useful for complementing a $DBpL$ resource description.

3.3.3 Comparing possible matches with reference data

When both DBp and $DBpL$ have been queried, the data collected for each candidate $c \in C$ is compiled by a separate Python module. First, regexes are used for extracting $Birthdate$ and $Club$ from each `dbo:abstract`(c). The extracted values are added to the other retrieved values $Birthdate(c)$ and $Club(c)$. Then, if additional data for c has been found in DBp, this is added to the description from $DBpL$. Table 2c demonstrates a compiled resource description of `dbr:Virgil_van_Dijk`, where the result from $DBpL$ has been complemented with values retrieved from DBp and the value "Liverpool" has been extracted from the abstract and added to $Club$(`dbr:Virgil_van_Dijk`).

After descriptions have been compiled, the candidates are compared with the reference player r, property by property. Another Python module was implemented for this task. This module includes various functions for comparing different properties. What was considered a match varied depending on the property's string representation in the reference dataset in relation to that retrieved from DBpedia.

We use the ρ notation also for properties of the reference players, i.e. $\rho(r)$ is the value of the property ρ of the reference player r. $Position(r)$ is represented in the reference dataset as either of the strings "Goalkeeper", "Defender", "Midfielder" or "Forward". $Position(c)$ of a candidate $c \in C$ can be strings such as "Association football midfielders" or "Forward (association football)". So in this context, it was considered a match when $Position(r)$ is a substring of any $Position(c)$ (There can be several values retrieved, only one needs to match).

Both $Number(r)$ and the numbers $Number(c)$ are numeric strings such as "9", "10" or "27", thus it was considered a match only if they are identical.

$Nationality(r)$ is represented as a country name string (e.g. "Germany"). The birth countries we find in DBpedia are also labeled with such strings,

Table 2: Descriptions of the player Virgil van Dijk queried from DBpedia. The "|" symbol separates different values associated with the same attribute. The abstracts have been shortened to fit the table.

(a) Description queried from DBpL.

URI	http://dbpedia.org/resource/Virgil_van_Dijk
Label	Virgil van Dijk
premPlayer	Yes
soccerPlayer	
Birthdate	
Number	
Nationality	
Position	Association football defenders
Club	
Abstract	Virgil van Dijk (born 8 July 1991) is a Dutch professional footballer who plays as a centre-back for Premier League club Liverpool and...

(b) Additional data queried from DBp.

URI	http://dbpedia.org/resource/Virgil_van_Dijk
soccerPlayer	Yes
Birthdate	1991-07-08
Nationality	Breda|Netherlands
Position	Centre-back

(c) Description after merging data from both DBpL and DBp and extracting data from abstract.

URI	http://dbpedia.org/resource/Virgil_van_Dijk
Label	Virgil van Dijk
premPlayer	Yes
soccerPlayer	Yes
Birthdate	1991-07-08
Number	
Nationality	Breda|Netherlands
Position	Centre-back|Association football defenders
Club	Liverpool

(d) Description after comparison with reference data.

URI	http://dbpedia.org/resource/Virgil_van_Dijk
Label	Virgil van Dijk
premPlayer	Yes
soccerPlayer	Yes
Birthdate	1991-07-08
Number	
Nationality	Netherlands
Position	Defender
Club	Liverpool

whereas the national team labels take the form of "Germany national football team", "France national under-20 football team", "Portugal Olympic football team" etc., with the country name in the beginning of the string. However, our queries implement the SPARQL built-in function `REPLACE` to remove trailing characters after the country name, thus *any* $Nationality(c)$ is returned as a plain country name string. Accordingly, when comparing $Nationality(r)$ with a country name $Nationality(c)$ in Python, only identical strings count as a match.

Birthdates are also exactly matched, however the strings required some preprocessing before they could be compared. $Birthdate(r)$ is in `dd/mm/yyyy` format, `dbo:birthDate`(c) are in `xsd:date` format, while a date in `dbo:abstract`(c) is in `dd Month yyyy` format. To enable exact matching, all *Birthdates* are converted to `xsd:date` format. The `strptime()` method from the built-in Python module `datetime` was used for this task.

Club names were considered a match when $Club(r)$ is a substring of a string $Club(c)$. Exact matching would not work in this case because most club strings in include prefixes/suffixes such as "F.C." or "A.F.C." (e.g. "Arsenal F.C."), whereas the reference club names either do not have a prefix/suffix at all (e.g. "Arsenal") or it does not include periods (e.g. "AFC Bournemouth"). $Club(r)$ is stripped off any prefix/suffix before the string comparison. A special case was when $Club(r)$ = "Brighton and Hove Albion". A regex then replaces $Club(r)$ so that both "Brighton & Hove Albion" and "Brighton and Hove Albion" can be matched.

When a candidate c has been compared with the reference player r, only the matched property values associated with the candidate are saved. The other values retrieved from DBpedia have played out their role at this stage of the algorithm. Table 2d shows the saved result after the property values in 2c have been compared with those in Table 1. In this case, all properties that had associated values matched. However, had e.g. "France" been retrieved instead of "Netherlands" as *Nationality*, the entry for this property would be empty, since neither "Breda" nor "France" match $Nationality$(Virgil van Dijk).

3.3.4 Choosing the best match

Finally, another module was implemented for systematically deciding which candidate c best matches the reference player r. A best practice for identifying LOD resources is not suggested in the literature, thus the system was created ad hoc.

A score is calculated for each c. The higher score, the better the match. Every matching property adds to the score a number $0 < \sigma \leq 1$ that is decided by the degree of uniqueness for a matched property. If δ is the number of resources sharing the property, then $\sigma = \frac{1}{\delta}$ is added to the total score. Properties shared by fewer resources entail a higher σ, whereas more generic properties add less to the score. For example, a matched birth date almost certainly adds more to the score than a matched nationality, since birth dates are generally more unique than nationalities. However, a very unusual country, linked to only a few resources, could theoretically generate a higher σ than a birth date if matched.

The number δ of resources sharing a property is queried at run-time. Template queries were constructed for this purpose. We let $S^{\rho(t)}$ denote the subset of resources S having the value $\rho(t)$ associated with property ρ. Then for a matched property value $\rho(c)$ of a candidate c,

$$\delta = \begin{cases} |L^{\rho(c)}|, & \text{for } \rho \in \{Number, Club\} \\ max(|L^{\rho(c)}|, |D^{\rho(c)}|), & \text{for } \rho \in \{Birthdate, Position, Nationality\} \end{cases}$$

A matched property value is inserted to the appropriate template and the query returns the number of resources having that property. The triple patterns in these queries correspond to those in the `OPTIONAL` clauses of the query that retrieved the property values in the first place (see Figure 10). Albeit now the searching is done in the opposite direction. Instead of starting from known URIs and searching for their properties, the queries start from the property value and search for URIs having this property. As an example, compare the two queries in Figure 11. The query in 11a initially searches for URIs matching the specified name Virgil van Dijk. It finds `dbr:Virgil_van_Dijk` and then searches for birth dates in the object position of triples having `dbr:Virgil_van_Dijk` as subject and `dbo:birthDate` as predicate. This is how the date "1991-07-08" was retrieved for `dbr:Virgil_van_Dijk` in the early stages of the algorithm. As the birth date later turned out to be a match, the number of resources having this birth date is looked up by the query in 11b. Here the birth date is inserted in the object position and the query searches for URIs in the subject position. The number δ is then returned using the `COUNT` function. In this case $\delta = 28$, meaning that the matching birth date contributes with $\sigma = 1/28 \approx 0.0357$ to the score for `dbr:Virgil_van_Dijk`.

```
SELECT DISTINCT ?Uri
   GROUP_CONCAT(DISTINCT xsd:date(?date);separator='|') AS ?Birthdate
{
   ?Uri rdfs:label ?Label .
   ?Label bif:contains "Virgil AND ('van' OR 'Dijk')"
   OPTIONAL {?Uri dbo:birthDate ?date}
}
```

(a) Find birth dates of matching URIs matching by name.

```
SELECT COUNT(DISTINCT ?Uri) AS ?bdCount
{?Uri dbo:birthDate "1991-07-08"^^xsd:date}
```

(b) Find URIs with the birth date 1991-07-08.

Figure 11: Retrieve birth dates and decide uniqueness of a matched birth date.

In addition to matching properties, being member of `dbc:Premier_League_players` or `dbo:SoccerPlayer` also contributes to the score. The δ values in these cases are constant for all candidates and reflect the number of members in either class, e.g. being one of the 4721 members of `dbc:Premier_League_players` contributes with $\sigma = 1/4721 \approx 0.0002$. Being a `dbo:SoccerPlayer` only adds to the score if the candidate is not a member of `dbc:Premier_League_players` too, since the latter implies the former.

For those properties $\rho(c)$ that are complemented with values from DBp, the queries that counts the corresponding δ are run against both datasets. Then δ

is set to the larger of the two results, as that should best reflect the degree of uniqueness. Optimally, δ would represent $|L^{\rho(c)} \cup D^{\rho(c)}|$, however no efficient way of calculating such a δ was found. Neither could resources with the property values contained in the abstract be counted. As a consequence, the query in 11b returned 0 in some cases. This happened when a $\mathit{Birthdate}(c)$ had been extracted from `dbo:abstract`(c) and no resources exist having the birth date represented with `dbo:birthDate`. In those cases, the birth date is counted as unique, thus adding $\sigma = 1$ to the total score.

When all σ values have been added together, a total score is given to the resource in question. The score is rounded to 4 decimals. The resource in the example, `dbr:Virgil_van_Dijk`, got a score of 0.0705. Having $\mathit{Birthdate}$ = "1991-07-08" and Club = "Liverpool" contributed with ~99.5% of the score. The remaining 0.5% came from being member of `dbo:Premier_League_players`, having $\mathit{Nationality}$ = "Netherlands" "and $\mathit{Position}$ = "Defender".

Ultimately, the candidate c with the highest score is picked and saved as the best match for the player r.

3.4 Measuring data quality

With a best matching resource identified for each player in R, data quality measurements could be carried out for this set of resources. As a consequence of discovering significant differences in their contents, both DBpL and DBp were assessed, despite the original intention being to only assess DBpL. The findings in Section 3.2 also played a part in choosing relevant measurements; syntactic correctness depends on predicate range and semantic correctness can only be decided if a predicate accurately correspond to the real-world property.

Some predicates that were certainly useful in the matching algorithm were considered too inaccurate or ill-defined to be included when measuring data quality. For example, birth country or national team is not the same thing as nationality. The natural way of expressing a person's nationality would instead be with `dbo:nationality`. This predicate is well-defined in the DBpedia ontology and frequently used ($|L^{dbo:nationality}| \approx 122000$), however not for football players. The predicates deemed most accurate for representing each property were:

- `dbo:birthDate` for $\mathit{Birthdate}$.
- `dbo:number` for Number.
- `dbo:position` for $\mathit{Position}$.
- `dbp:currentclub` for Club.
- `dbo:nationality` for $\mathit{Nationality}$.

Population completeness was determined by the result returned by the matching algorithm. For a resource to be considered a valid match, a score of at least 0.001 was required. A score of 0.001 corresponds to having matched by name plus at least one other property and that property being shared by at most 1000 resources. Manual look-ups of the resources with the lowest scores above 0.001 was conducted to ensure correctness. We define the set of valid matches as V.

The population completeness in $DBpL$ could then be measured as the ratio $\frac{|V|}{|R|}$. A corresponding metric for DBp was produced by checking how many of the matching resources also existed in DBp, i.e. the ratio $\frac{|V \cap D|}{|R|}$.

Property completeness was measured for each property respectively. Property values $\rho(v)$ of each $v \in V$ was queried using only the valid predicates, with V inserted in the query as `VALUES`. The respective metrics could then be extracted from the query result set. For each property ρ, the completeness in $DBpL$ was calculated as the ratio $\frac{|V^{\rho}|}{|V|}$. For example, if there are 100 matched resources and 33 of them have values associated with `dbo:birthDate`, the completeness of $Birthdate$ in $DBpL$ is 33%. Property completeness in DBp was measured as $\frac{|(V \cap D)^{\rho}|}{|V \cap D|}$.

Syntactic accuracy was measured only for $Birhdate$ and $Number$, since `dbo:birthDate` and `dbo:number` were the only two predicates having typed literal ranges, `xsd:date` and `xsd:string` respectively. Literals for $Position$, $Club$ and $Nationality$ are all provided in triples with `rdfs:label`. With the range of `rdfs:label` being any literal type, relevant measurements for those literals could not be accomplished. When querying properties of all $v \in V$, datatypes of the property values in $Birthdate(v)$ and $Number(v)$ were selected using the `DATATYPE` function in SPARQL. Syntactic accuracy could then be calculated as the percentage of retrieved values having correct datatype.

Semantic correctness of property values $\rho(V)$ was determined by comparison with the property values in R. A property value $\rho(v)$ of a $v \in V$ was considered semantically correct if it matched the corresponding $\rho(r)$. The criterias for being a match were the same as those described in section 3.3.3. Semantic accuracy was then determined by the proportion of property values being semantically correct.

4 Results

This section presents the results of the study. First, the data quality measurements – presented as tables – are analyzed. The subsequent discussion is split in two parts: the first discusses the utility of DBpedia's football data and the second emphasizes the role of ontologies.

4.1 Analyzing the results

As can be seen from Table 3, most EPL players are represented in *DBpL*. One would expect this, since DBpedia resources correspond to Wikipedia pages, and Wikipedia is well-known for covering a lot of topics. That *DBp* has less player coverage is also natural, with many new players emerging in the league in recent years.

The numbers for population completeness are based on the matching algorithm result. In order to validate this result, 20 unmatched player names were looked up in Wikipedia with free-text search. For 14 of the 20 players, Wikipedia pages with corresponding DBpedia resources were found. This indicates that the *actual* population completeness might be significantly higher than suggested in Table 3.

Common to most player resources that the matching algorithm failed to find is that their names contain diacritics that are not present in corresponding reference names. E.g., it fails to find the resource with `rdfs:label` "Muhamed Bešić" due to the reference name being spelled "Muhamed Besic". One way of working around this problem, and thus obtain a more accurate metric for population completeness, would be to use several different sources for reference. Another way would be to set up a local DBpedia mirror with Virtuoso and then configure the `bif:contains` function to be "diacritic insensitive" (OpenLink Software 2016). However, the latter option seems unavailable for DBpedia Live at the time of writing (DBpedia 2019). Matching by another property than name would also have been a feasible solution, should the property be well-represented among player resources in *DBpL*. The more properties with a high degree of completeness, the more options for identifiers, thus identification does not stand or fall with a specific property being represented in a certain way.

However, the numbers in Table 4a clearly shows that all of the assessed properties have poor representation in *DBpL*. Properties of a player resource are much likelier to be found in *DBp*, as is demonstrated by Table 4b. It can also be seen that any property can not be retrieved. None of the player resources in either dataset have their nationality described. This is most likely due to DBpedia extracting most data from Wikipedia page infoboxes and these do not include nationality. Current club is not described for any player in *DBp*

Table 3: Population completeness.

Dataset	**Matched resources**	**Completeness %**
DBpL	550	88.5
DBp	400	64.3

Table 4: Property completeness.

(a) Property completeness in DBpL.

Property	**Resources with property**	**Completeness %**
Birthdate	32	5.8
Number	30	5.5
Position	30	5.5
Club	30	5.5
Nationality	0	0

(b) Property completeness in DBp.

Property	**Resources with property**	**Completeness %**
Birthdate	366	91.5
Number	360	90
Position	366	91.5
Club	0	0
Nationality	0	0

either, despite being something that is displayed in Wikipedia infoboxes. This is however reasonable, since this property could quickly be outdated. The club of the player at the time is more likely described by `dbo:careerStation` than `dbp:currentclub` in *DBp*.

In terms of syntactic accuracy, it seems that DBpedia can trusted. Table 5 shows that all assessed values conform to the data type range of their associated predicates.

Birth date stand out as the property with the highest semantic accuracy, as can be seen from table 6. There are several explanations for this.

Firstly, RDF-statements expressing birth date can never be outdated, whereas those expressing e.g. number become semantically incorrect if the real-world player changes number. As demonstrated by 6b, numbers have low accuracy in *DBp*, which is likely due to most being outdated.

Secondly, it goes without saying that a resource can only have one correct birth date. E.g. positions on the other hand can have different correct values. The real-world player might play as a defender in some games and as a midfielder in others. For resources being associated with several distinct positions, only one position was counted as correct even if the other are also semantically correct, since the reference players are only associated with one position each. There is also a possibility that semantically correct values for number or club were erroneously counted as incorrect, due to the DBpedia ontology not defining what is expected of values associated with `dbo:number` or `dbo:currentclub`. It was assumed that `dbo:number` refers to the *current* number, however it could be the case that it refers to a player's number at any time, hence different values could be correct. While `dbp:currentclub` probably speaks for itself, it can not be ruled out that different associated values can be semantically correct, should the predicate refer to *any* type of club and not just football clubs. Still, outdated statements is a more feasible explanation for resources having incorrect clubs.

A third reason for birth dates having superior semantic accuracy is that `dbo:birthDate` has a defined range of `xsd:date` literals, and since all DBpedia-retrieved birth dates were syntactically correct, they could be properly compared with the reference dates. Comparing positions was not as straight-forward. Again due to lacking ontology definitions, some semantically correct values for position were possibly not counted as such. The reference positions are limited to either of goalkeeper, defender, midfielder and forward, whereas in DBpedia many other positions such as right-back, winger or striker appeared in triples with `dbo:position`. Even though e.g. right-back is a type of defender, the former was not counted as semantically correct when compared with the latter. This problem could have been avoided, had the range of `dbo:position` been defined. Then ad hoc mappings from each position in the range to either of the four reference positions could have been implemented. Even better would be if the positions were classified already by the ontology, e.g. right-back being subclass of the defender being subclass of football positions etc. Then positions could be queried by class instead, which would facilitate a more rigid assessment of semantic accuracy.

Table 5: Syntactic accuracy.

(a) Syntactic accuracy in DBpL.

Property	**Property values**	**Correct datatype**	**Accuracy %**
Birthdate	34	34	100
Number	37	37	100

(b) Syntactic accuracy in DBp.

Property	**Property values**	**Correct datatype**	**Accuracy %**
Birthdate	370	370	100
Number	360	360	100

Table 6: Semantic accuracy.

(a) Semantic accuracy in DBpL.

Property	**Property values**	**Correct values**	**Accuracy %**
Birthdate	34	31	91.2
Number	37	25	67.6
Position	34	22	64.7
Club	35	27	77.1
Nationality	0	0	-

(b) Semantic accuracy in DBp.

Property	**Property values**	**Correct values**	**Accuracy %**
Birthdate	370	362	97.8
Number	360	171	47.5
Position	404	310	76.7
Club	0	0	-
Nationality	0	0	-

4.2 Discussion

The discussion considers two parts; DBpedia as a source for football data, and the importance of ontologies.

4.2.1 DBpedia as a source for football data

The data quality measurements showed deficiencies in DBpedia's football data, with the most obvious being the lack of properties associated with EPL players in *DBpL*. One of the strengths with linked data is that conclusions can be drawn about entities of a particular type. Poor property completeness undermines the validity of such conclusions. It seems thus that *DBpL* should only be used for querying information about the individual player, though using DBpedia this way hardly has any advantage over traditional web browsing.

The non-live version of DBpedia should be better for querying information that involve data from many resources, although the set of retrievable properties seems to be limited to those present in Wikipedia infoboxes. The temporal aspect also needs to be taken into account. Due to its outdatedness, *DBp* is not guaranteed to include most entities of a queried type and mutable properties are likelier to be incorrect.

As was proven when implementing the matching algorithm, a more complete representation of the players could be obtained by combining data from the two datasets. Filtering such as "find all players with number 9" should be reliable thanks to good syntactic accuracy and the statements should in most cases be semantically correct. However, even when combining the two datasets, a signicantly large amount of player resources still have incomplete descriptions, making it difficult to extract credible information about the players as a group.

If EPL players are representative for football-related resources in DBpedia in general, little suggest that DBpedia data is of use in the football domain. However, it was possible to at least identify most players in DBpedia. Should more comprehensive football data exist elsewhere in the LOD cloud, DBpedia could be a possible starting-point for accessing it. Wikipedia is unparalleled in covering different topics, and by identifying the topic of interest, one can query the corresponding DBpedia resource for outgoing links to other datasets. Of the 550 players found in *DBpL*, 337 were provided with the `owl:sameAs` predicate, thus there might be links to more information about them.

4.2.2 The importance of ontologies

Apart from DBpedia's football data itself being of seemingly poor quality, DBpedia also turned out to be deficient as a football ontology. With many of the ontology terms being ill-defined, several assumptions had to be made about them. Assumptions about ontology terms were based on the data instances, although it should be the other way around. Much effort that was spent on understanding the relations in DBpedia could have been avoided, had the ontology been clearer. Moreover, a better defined ontology would facilitate data quality assessment.

According to Zimmermann (2010, p. 95) a lack of appropriate ontologies is a problem for Linked Data publishers. He claims that there are too few well-known ontologies, hence it can be a difficult task finding appropriate terms to describe data within a domain. Transformation of data in traditional databases into Linked Data to a greater extent would be encouraged if domain-specific ontologies were more developed. Thus, the apparent under-representation of football data as LOD could be due to the absence of a football ontology.

A typical problem that Zimmermann (2010, p. 95) mentions is that ontologies "exist but are difficult to find because developed by small groups for experimentation, lacking advertisement". This appears to apply to the work of Bergmann et al. (2013) who have made an effort to create what they claim to be the first comprehensive football dataset published as Linked Data. Their dataset consists of more than 9 million RDF-triples describing various football concepts and is accompanied by *Soccer Voc*, a dedicated football ontology created by themselves.

The work of Bergmann et al. (2013) could potentially pave way for more football data being published as LOD. However, it seems to have disappeared from the web. This could be due to restrictions from organizations owning the rights to the data. Bergmann et al. (2013) describe sports databases as proprietary and only published a fraction of their created dataset in the first place, due owner rights.

5 Conclusions

The World Wide Web constitutes a seemingly endless source of information, although fractured and dispersed. Machines can help processing large amounts of data. Linked Open Data takes advantage of the opportunity that the Web represents, while at the same time using a reasonable amount of computing resources.

Linked open datasets grow steadily, but this growth is not even across all domains. As an example, the football domain is well-represented in terms of resources and applications in the "normal" Web, but lacks of dedicated dataset and ontology in Linked Open Data.

With this study we wanted to assess the status of the linked open football data and understand the reasons behind its lack of development. We selected DBpedia as the largest cross-domain dataset available, and Premier League data as reference sample. We began by exploring the DBpedia ontology for terms describing football resources, and with these, we built an algorithm to identify and extract a set of Premier League resources. Finally we analyzed this set with the help of established quality criterias and discussed the results.

The study of the football data in DBpedia shows that without schemas declaring the expected appearance of the data, its extraction requires more specific domain knowledge, e.g. to design queries and validate the data. The advantage of DBpedia is that the amount of resources grows continuously. This is also shown in the football domain (population completeness). However those resources often lack comprehensive descriptions (property completeness), limiting the use of DBpedia's football data in its current form.

Despite this limitation, the fact that DBpedia is connected to other data sets opens up the opportunity to investigate its role as a stepping stone to extend the descriptions with data from other sources.

The challenges occurring in this study have highlighted the importance of well-designed ontologies. The football ontology, rather the lack thereof, limits to a great extent the development of the domain. Thus, it would be of importance to analyze the factors that would stimulate this development, like economic and social value of making available a complete linked open football dataset.

References

Addlesee, Angus (Oct. 31, 2018). *Creating Linked Data.* URL: https://medium.com/wallscope/creating-linked-data-31c7dd479a9e. (accessed: 2020-04-28).

Bergmann, Tanja et al. (2013). "Linked Soccer Data". In: *Proceedings of the I-SEMANTICS 2013 Posters & Demonstrations Track*, pp. 25–29.

Berners-Lee, Tim (July 27, 2006). *Linked Data - Design Issues.* URL: https://www.w3.org/DesignIssues/LinkedData.html. (accessed: 2020-04-02).

Bizer, Christian, Tom Heath, and Tim Berners-Lee (2009). "Linked Data - The Story So Far". In: *Int. J. Semantic Web Inf. Syst.* 5, pp. 1–22.

Brickley, Dan and Ramanathan Guha (Feb. 2014). *RDF Schema 1.1.* W3C Recommendation. W3C. URL: https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

DBpedia (Dec. 15, 2009). *DBpedia Community.* URL: https://wiki.dbpedia.org/about/dbpedia-community. (accessed: 2020-08-28).

— (Jan. 26, 2018). *Public SPARQL Endpoint.* URL: https://wiki.dbpedia.org/public-sparql-endpoint. (accessed: 2020-05-25).

— (July 3, 2019). *DBpedia Live.* URL: https://wiki.dbpedia.org/online-access/DBpediaLive. (accessed: 2020-05-18).

DCMI Usage Board (Jan. 2020). *DCMI Metadata Terms.* DCMI Recommendation. Dublin Core Metadata Initiative. URL: https://www.dublincore.org/specifications/dublin-core/dcmi-terms/.

Dean, Mike and Guus Schreiber (Feb. 2004). *OWL Web Ontology Language Reference.* W3C Recommendation. W3C. URL: https://www.w3.org/TR/2004/REC-owl-ref-20040210/.

DuCharme, Bob (2011). *Learning SPARQL.* O'Reilly Media, Inc. ISBN: 1449306594.

Heckmann, Dominikus (2006). "Ubiquitous User Modeling". PhD thesis.

Matuszka, Tamás and Attila Kiss (2014). "Alive cemeteries with augmented reality and semantic web technologies". In: *International Journal of Computer, Information Science and Engineering* 8.2, pp. 32–36.

McCrae, John P. (Mar. 29, 2019). *The Linked Open Data Cloud.* URL: https://lod-cloud.net/. (accessed: 2020-04-02).

Miles, Alistair and Sean Bechhofer (Aug. 2009). *SKOS Simple Knowledge Organization System Reference.* W3C Recommendation. W3C. URL: https://www.w3.org/TR/2009/REC-skos-reference-20090818/.

OpenLink Software (Sept. 9, 2016). *How Can I Control the normalization of UNICODE3 accented chars in free-text index?* URL: http://docs.openlinksw.com/virtuoso/virtuosotipsandtrickscontrolunicode3/. (accessed: 2020-05-18).

— (2020). *Virtuoso SPARQL Query Editor — Namespace Prefixes.* URL: http://live.dbpedia.org/sparql?help=nsdecl. (accessed: 2020-08-22).

Rula, Anisa, Andrea Maurino, and Carlo Batini (Mar. 2016). "Data Quality Issues in Linked Open Data". In: pp. 87–112. ISBN: 978-3-319-24104-3. DOI: 10.1007/978-3-319-24106-7_4.

Seaborne, Andy and Steven Harris (Mar. 2013). *SPARQL 1.1 Query Language.* W3C Recommendation. W3C. URL: https://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

SPARQLWrapper Documentation (May 6, 2020). URL: https://readthedocs.org/projects/sparqlwrapper/downloads/. (accessed: 2020-08-26).

Vega-Gorgojo, Guillermo (2019). “Clover Quiz: a trivia game powered by DBpedia”. In: *Semantic Web* 10.4, pp. 779–793.

W3C (Oct. 25, 2009a). *Semantic Web*. URL: https://www.w3.org/standards/semanticweb/. (accessed: 2020-04-02).

— (Dec. 11, 2009b). *Vocabularies*. URL: https://www.w3.org/standards/semanticweb/ontology. (accessed: 2020-04-26).

— (Aug. 29, 2012). *W3C Mission*. URL: https://www.w3.org/Consortium/mission. (accessed: 2020-04-22).

Zaveri, Amrapali et al. (2016). “Quality assessment for linked data: A survey”. In: *Semantic Web* 7.1, pp. 63–93.

Zimmermann, Antoine (2010). “Ontology recommendation for the data publishers”. In: *ORES-2010 Ontology Repositories and Editors for the Semantic Web*, pp. 95–99.