# Basketball Question Answering using Knowledge Graphs and Large Language Models*

Michalis Mountantonakis[1,2][0000−0002−1951−0241], Dimitris Papadopoulos[2][0009−0003−7630−3569], and Yannis Tzitzikas[1,2][0000−0001−8847−2130]

[1] Information Systems Laboratory, FORTH-ICS, Heraklion, Greece
[2] Computer Science Department, University of Crete, Heraklion, Greece
mountant@ics.forth.gr, csd4976@csd.uoc.gr, tzitzik@ics.forth.gr

**Abstract.** There is a recent trend of using LLMs for generating SPARQL queries from natural questions, in order to exploit the generated query for retrieving the answer over a Knowledge Graph (KG). This can be quite challenging for popular domains, such as sports, that include billions of fans worldwide that are not familiar with KGs and query languages, and they typically express their questions in natural language. In this paper, we focus on basketball games which are complex events including several factors and participants, such as teams, players, statistics and others. However, all this information is usually given in separate HTML tables or JSON files, thereby, it is not trivial to be combined. The objective is to show a generic pipeline including the following steps: i) data collection, ii) the definition of competency questions, iii) the creation of an event-based ontology and the KG, and iv) the exploitation of the KG through the LLMs, for enabling natural QA through a text-to-SPARQL method based on ontology and literals path patterns. As a use case, we construct a KG including 5 million triples from the EuroLeague games from 2000-01 until 2024-25, a web application, and an evaluation benchmark with 100 natural questions in 3 languages (English, Greek, Chinese). Finally, we present an experimental evaluation for the QA task; indicatively by using the proposed method, we reached 0.79 accuracy through GPT-4 and 0.71 accuracy through DeepSeek, for the English benchmark, instead of 0.3 for the baseline method.

**Keywords:** Basketball Analytics · LLMs · Knowledge Graphs · QA

## 1 Introduction

In several real world domains, such as lifesciences, geography and others [15], Knowledge Graphs (KGs) are constructed for facilitating data analysis and offering advanced services, by answering more complex questions that require to

* This is a preprint of the paper: Michalis Mountantonakis, Dimitris Papadopoulos and Yannis Tzitzikas, Basketball Question Answering using Knowledge Graphs and Large Language Models, Accepted for Publication in the in-use track of the 14th International Joint Conference on Knowledge Graphs (IJCKG 2025)

combine knowledge from several entities. Towards this direction, the high trend of using Large Language Models (LLMs) has resulted in the creation of several approaches that try to exploit LLMs for generating SPARQL Queries for a given natural question over a desired KG [14,3,18], e.g., see an example for the Cultural Heritage domain [18]. However, most web users are not familiar with KGs and their related technologies. As a result, users typically express their questions in natural language, and even in their native language.

Concerning sports domain, it interests a high number of people worldwide, and regarding research, sports analytics is an emerging topic as it is stated in [23,5]. Here, we focus on Basketball, which is a very popular sport worldwide with over 3 billion fans (`http://bit.ly/3QK136X`), and over 600 million fans in China. Each Basketball game is a quite complex event including several participants and factors, such as players, coaches, referees, audience, statistics and others. As a result, there is a lot of information that needs to be recorded and there is a high need for updates, since the action never stops; new games are played all the time, new players join the teams, etc. Moreover, there are numerous simple and complex questions, that can be asked either from the fans and journalists for retrieving specific facts, or from the coaching staff for performing a data analysis. For example, to optimize the performance of players and to enable the decision making [27], e.g., a recent study by MIT researchers [29] states the high influence of basketball analytics investment on team performance.

Generally, there are numerous webpages offering traditional and advanced sports statistics in HTML tables and/or in JSON format derived by relational databases. However, there is a lack of RDF KGs for sports domain including such statistics, e.g., the pioneer researcher G. Weikum has stated [30] that "Capturing quantity properties of entities in sports, is important for KG coverage towards analytic tasks. This is a big gap in today's major KGs, and search engines are not a good proxy". Therefore, a major challenge is (a) to construct such RDF KGs for sports for enabling the answer of complex questions, such as those involving relationships across multiple entities (players, coaches, referees, countries), semantic constraints, and numerical statistics, and (b) to offer natural Question Answering (QA) by translating each question into precise SPARQL expressions. Indeed, SPARQL can allow sports analysts to efficiently calculate aggregated metrics such as average, total, max and min directly from large KGs without manually processing individual game records.

As a running example, suppose that a fan of European Basketball desires to find an answer for the question: "Give me the games that each of Kendrick Nunn and Kostas Sloukas, as teammates, scored at least 15 points". To answer this question, the classical approach is to browse every Panathinaikos game boxscore, to find the required information. And this can become even harder for questions that one should combine data from more than one teams, seasons, statistics, players, and others, or/and specific characteristics of them, such as "Is there any game that a player of a Greek team, having height less than 2.00, that had over 10 rebounds and over 10 points." On the other hand, the proposed approach (see Fig. 1) is to exploit a KG where one can find and combine all the

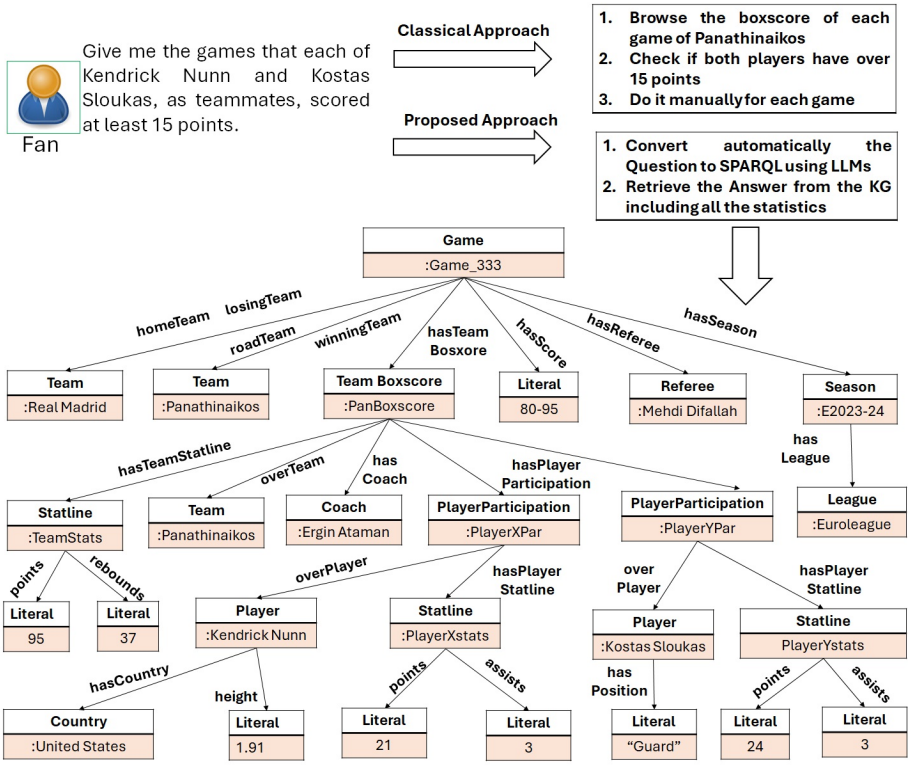


Fig. 1: A Query from a user and the Knowledge Graph for a Basketball Game

information in a structured way, and then to enable the conversion of natural language questions to SPARQL, for answering such questions, even in his/her native language. Below we present the research questions:

- *RQ1:* How can we design and construct a domain-specific ontology and KG to support natural QA and complex analytics tasks in basketball?
- *RQ2:* How effectively can LLMs generate accurate SPARQL queries from natural language questions in the basketball domain, when guided by the ontology paths and literal patterns?

Regarding our contribution, in this paper i) we describe a pipeline including all the steps, from the data collection, to the KG exploitation using LLMs, ii) we propose an event-based ontology for basketball games, called *BBall*, and we construct a KG for basketball games by focusing on EuroLeague Basketball, and iii) we use a text-to-SPARQL method based on ontology path and literals patterns that are given to LLMs for enabling natural QA over the KG and we have built an application called *HoopProphets*, which is available online in `https://demos.isl.ics.forth.gr/HoopProphets`. Finally, iv) we present comparative evaluation results by constructing a benchmark with 100 natural

questions (that also includes a paraphrased version in English, a Greek and a Chinese version), by using popular LLMs, i.e., ChatGPT and Deepseek.

To the best of our knowledge, this is the first work proposing such a pipeline, including all the steps from the ontology creation to the KG exploitation from the LLM for enabling the expression of natural language questions. Moreover, we present a KG for the most popular European Basketball league, that can be exploited for QA, data analytics and insights, game simulations and others.

The rest of this paper is organized as follows: §2 discusses related work, and §3 describes the pipeline and all the steps. Furthermore, §4 presents the experimental results based on a QA benchmark, and finally §5 concludes the paper and discusses the directions for future research.

## 2 Related Work

Here we discuss the: i) Generation of SPARQL from texts, and ii) KGs and analytics for basketball. Finally, we provide a comparison with our work.

### 2.1 SPARQL Generation from text

First, the authors in [32] exploited Chinese KGs for SPARQL generation over LLMs, whereas in [25], example SPARQL queries and question analysis techniques combined for SPARQL generation over the Open Research KG. In [3], several configurations were compared for SPARQL query generation (such as few shows and fine tuning) for the DBpedia [1] and Wikidata [28] KGs. Also, the authors in [2] proposed a hybrid approach for text-to-SPARQL by combining LLMs with RAG techniques, while [14] evaluated the performance of fine-tuned LLMs for the generation of SPARQL queries and analyzed its limitations. Moreover, in [24], a curated dataset with 2,771 unique queries from Wikidata was released, which can be used for fine-tuning LLMs to generate accurate and syntactically valid SPARQL queries from natural text. Finally, in our past work [18], we have proposed ontology path patterns methods, for generating complex queries over the ISO standard CIDOC-CRM for Cultural Heritage domain KGs.

### 2.2 KGs and Analytics for Basketball

First, the authors in [9], collected data about teams and players from basketball websites and created a KG about NBA, and used BERT for performing QA over the KG. Also, in [10] a QA system based on an NBA KG was proposed, which included a combination of NLP and Machine Learning techniques for answering questions. Another work [13], analyzed and compared statistical data of NBA and EuroLeague Basketball, and they mentioned that EuroLeague "is becoming quantitatively and qualitatively more similar to the NBA". Additionally, there are also works for the prediction of the winner of a basketball game based on machine learning algorithms, for EuroLeague [8] and NBA [26]. Regarding statistics, in [7] a statistical analysis was performed on the historical evolution of

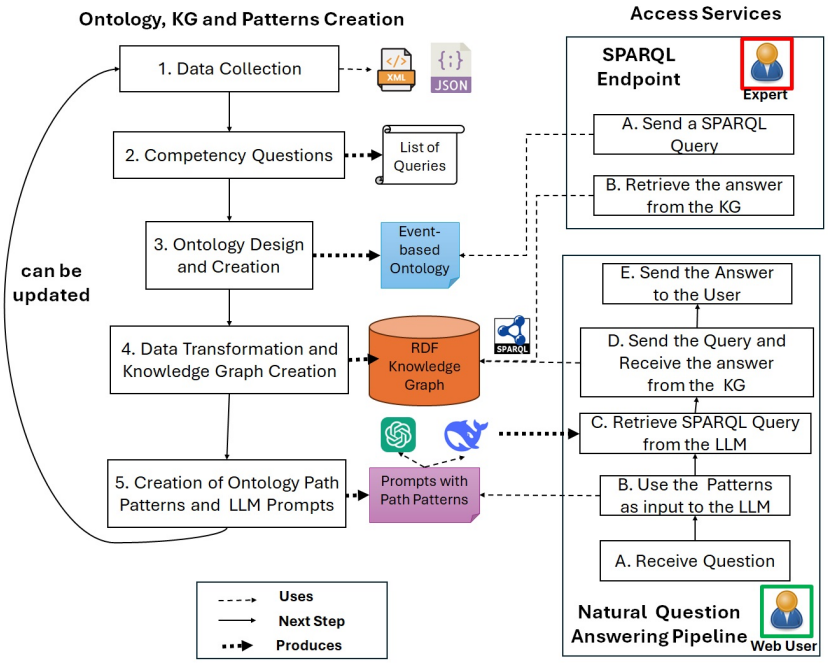


Fig. 2: A Pipeline for Enabling Natural QA with LLMs over Event-Based KGs

EuroLeague. In addition, in [6], LLMs were used for evaluating several tasks for Basketball analytics, such as for analyzing critical statistics in game summaries, whereas [22] used association rule mining techniques for evaluating recovery and economic impacts in NBA basketball. Finally, in [31] the benchmark SportQA was presented, which contains 70K questions for several sports (including basketball) and subjects, such as historical facts and complex scenarios.

### 2.3 Comparison and Novelty

Compared to the basketball analytics (e.g., [9,10]), this is the first work i) providing a pipeline for basketball games including all the steps from the production of the KG to its exploitation through LLMs for enabling natural QA, and ii) offering a KG and a QA application for the EuroLeague basketball. Regarding text-to-SPARQL, similarly to our past work [18], we also exploit ontology path patterns methods (but here we provide a more compact version by also including literal datatypes), and not fine-tuning or few shots techniques (e.g., [3,14]).

## 3 The Proposed Steps of the Pipeline

Here we describe all the steps of the pipeline, which are depicted in Fig. 2. As we can see, in the left part, the key steps are the data collection, the creation

of Competency Queries, the design and implementation of the ontology, the data transformation and KG creation, and finally the creation of path patterns for constructing prompt templates. On the right part, we can see the steps for exploiting the resulted prompt templates for SPARQL Query Generation over the KG, for offering advanced query services for natural text questions.

### 3.1 Data Collection

The first step is the data collection, which is important for defining the competency queries and for designing the ontology. Here, we exploit the EuroLeague API[3], where one can export several information about EuroLeague Basketball in JSON or XML format. However, the data are not inherently connected, as they do not follow Linked Data principles, requiring additional processing to establish relationships. Thereby, it is not trivial to ask questions that require information from several API requests. We use 9 endpoints from the API for retrieving in JSON format a) the key metadata of the game, b) the game statistics, and c) metadata for players, coaches, referees, teams, countries and venues.

### 3.2 Competency Queries

We provide 5 indicative competency questions (in 5 categories) that are not trivial to be answered by the classical boxscores. The full list is available in `https://github.com/mountanton/BasketballQA`.

**Q1. General Question.** Give me the winning percentage of Ergin Ataman in 2022-23.

**Q2. Filtering Question.** Give me all the games of 2023-24 in which the losing team had a higher Performance Index Rating than the winning team.

**Q3. Top-K Question.** Give me all the top-10 players with the most points in a game of 2023-24 having height less than 1.85.

**Q4. Grouping Question.** Show Nigel Hayes-Davis's monthly total rebounds in 2023–24.

**Q5. Negation Question.** Give me the number of teams that have never won Barcelona at road since 2020.

### 3.3 Ontology Design and Creation: The BBall Ontology

Although basketball is a very popular sport, we did not manage to find an ontology to cover our competency questions. The most related one is the IPTC Sports Schema[4], however, it mainly contains generic properties for basketball statistics and cannot record the required information of a basketball game. For this reason, we decided to create an ontology called *BBall* that describes the key statistics of a Basketball Game. As we can see in Fig. 3, the key class is the *Game*, which is an *Event* and is connected with the *Teams*, the *Venue*, the

[3] `https://api-live.euroleague.net/swagger/`
[4] `https://sportschema.org/ontologies/basketball`

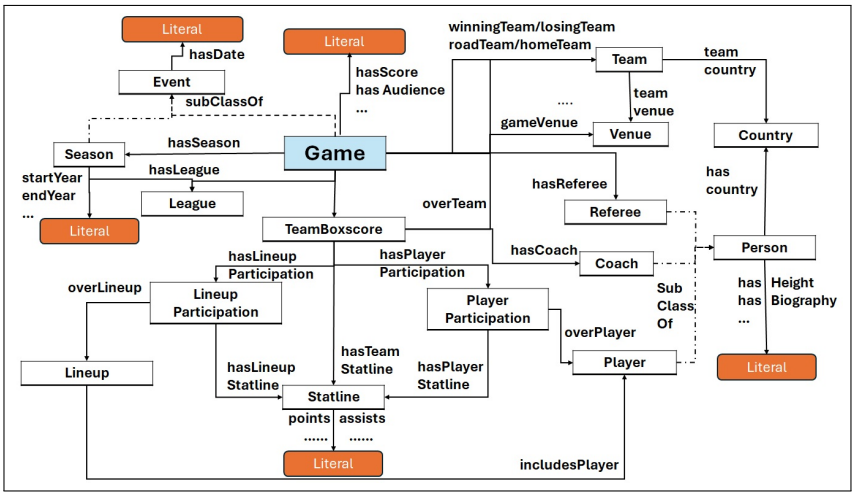


Fig. 3: The BBall Ontology - Key Classes and Relationships

*Referees*, the *Season*, and contains data properties for the score, the audience, etc. It is also connected with the *TeamBoxscore*, which contains the boxscore of each of the 2 teams of the game. First, each boxscore is connected with a *Team*, and the *Coach* of the team, and also it has a team *Statline*, i.e., for recording the total statistics of the team in the given game. The *Statline* class has one property for each statistic of a boxscore, such as points, rebounds, etc.

Regarding *Players*, we have the *PlayerParticipation* class, which records the participation of each player in the game. For each player participation, we need also to record his *Statline*, i.e., to cover all the statistics of the player. Also, the ontology has another one class related to the statistics, i.e., the *LineupParticipation*, in case that one desires to provide the statistics of a given K-player lineup, i.e., the statistics of the team when a given 5-player lineup is on the court, e.g., see [16]. Finally, for each of the other classes there are several properties, such as for the *Person* (e.g., Players, Coaches) the name, height, birthdate and *Country*; for the season the corresponding *League*, the start year, and so on. An example of the ontology using real data is depicted in Fig. 1, where we start from a game, and we can see the teams, a boxscore of a team, statistics, players, etc.

**Validation, Documentation and Publication of the Ontology.** For validating the consistency, completeness and conciseness of the BBall ontology we used the tool "Oops!" [21], and the ontology passed the tests without critical errors. Moreover, we use WIDOCO [4] for the documentation of the ontology, which also provides an interactive visualization through the WebVOWL system [12]. Its documentation and the ontology are available online[5]. The ontology contains 16 classes, 25 object properties and 63 data properties (most of them are not depicted in Fig. 3 for reasons of space). Finally, after the ontology creation, an important task is to construct the SPARQL query for each competency question.

[5] `http://users.ics.forth.gr/~mountant/bball.html`

```
SELECT DISTINCT ?class ?p (?object AS ?classOrDatatype)
WHERE {
  ?s ?p ?o . ?s a ?class .
  OPTIONAL { ?o a ?object . }
  OPTIONAL { ?s ?p ?o. FILTER(isLiteral(?o)) .
    BIND(DATATYPE(?o)  AS ?object)
  }
} ORDER BY ?class ?p
```

Fig. 4: The SPARQL query for retrieving the path patterns

**Possible Extensions.** Certainly, the ontology can be extended to cover more statistics and more sub-events of the game, such as the play-by-play events and more advanced statistics. However, it is worth noting that most of them, such as the number of points per minute, assists/turnover ratio, and others, can already be computed by exploiting the aggregation capabilities of SPARQL (e.g., using COUNT, SUM, AVG, MAX, MIN, etc.). Additionally, more properties can be added for several classes, such as for persons, venues, teams and countries.

### 3.4 Data Transformation and KG Construction

By using the *BBall* ontology, we need to transform the data and one key objective is the URI policy for having both dereferencable URIs and a single URI for each real world entity. Regarding the steps, they are explained below:

• We retrieve the metadata of all the games for each season. Then we retrieve each game and we create the RDF triples using the *BBall* ontology, and some key properties from the RDF, RDFs, SKOS and FOAF, for the game metadata and the statistics for the players and the team.

• For each person, the teams, venues and countries, we also store their IDs. we exploit the endpoints offered by the API for the persons, teams and venues, for retrieving the data and for transforming them in RDF format.

• In the end, we have created a single ".ttl" file for the games of each season, all the players, teams, referees, coaches, venues and countries, a total of 175 ".ttl" files. Each of the real world entities includes a unique URI having as a suffix a unique code. For most of the entities, we offer dereferencable URIs. The produced files in Turtle format, which have size 322 MB on disc, have been uploaded in a SPARQL endpoints by using the Virtuoso v7 engine.

### 3.5 QA Access Services: Path Patterns Creation and LLM prompts

The final step is to create ontology path and literal patterns as input to the LLM for enabling SPARQL Query Generation. The key advantages of such an approach are the following: a) it can be used for non-familiar users to SPARQL, for expressing any question that even require to combine information from multiple paths, b) it is a generic method that can be applied in any event-based RDF KG without a need to train the whole KG using an LLM, and c) it can support

multilinguality, i.e., the question can be expressed in any language. For enabling the natural QA over the KG, one needs first to create the patterns through a generic SPARQL query. Compared to [18], in this paper we have an ontology which is not so complex as CIDOC-CRM, i.e., in CIDOC-CRM one can use different path patterns to express the same events. For this reason, here we propose to have a single compact prompt template, where we provide the ontology path patterns grouped by each RDF class, and we also include the datatypes of literals (e.g., they can be important for queries that require sums, averages, comparisons, etc.). Fig. 4 shows the query to retrieve the path patterns, including patterns having either as an object a class, such as "Game–>homeTeam–>Team", or a literal, such as "Game–>hasHomeTeamScore–>Literal (xsd:integer)".

The next step is to create the prompt templates using the mentioned ontology path and literals patterns, and optionally, to include extra data patterns, such as URI patterns about the suffixes, and/or Literals patterns, like patterns related to spaces, capitalization and others. For each triple pattern, we distinguish two types of objects for also enabling the visit of longer paths (e.g., for multi-hop queries): a) for the classes we use the "*go to*" phrase for showing the class that we need to visit for constructing a longer path and b) for literals, we give its datatype. Fig. 5 presents such a prompt, where the patterns are grouped by class and extra guidelines are provided for patterns related to the URIs and literals (e.g., person names, team names, URI policy).

**Natural QA by using Path Patterns Prompts.** The notion is first to retrieve the given question $Q$, and then to exploit the pre-generated prompt template(s), i.s., right part of Fig. 2. A prompt is sent to the desired LLM, which generates the SPARQL query based on the prompt guidelines. The next step is to send the query to the SPARQL endpoint and to retrieve the answer. In cases where the SPARQL query returns an empty set, we can repeat the last two steps. Finally, the results can be either shown in HTML tables and/or can be sent to an LLM to create a more natural answer. We can use the same prompt for any language, by sending the question $Q$ expressed in the desired language.

**Application.** The application *HoopProphets* offers the multilingual natural QA over the KG in `https://demos.isl.ics.forth.gr/HoopProphets`. A screenshot with an example of a complex query is shown in Fig. 6. Also, a tutorial video can be found in `https://youtu.be/X6HCxxFDgXM`.

**Additional Steps: Updates and Enrichment.** Regarding the updates, the steps (or some of the steps) should be repeated in many cases, such as for adding new data like new basketball games, or/and new types of data (such as the play-by-play) which will maybe result in the extension of the ontology. Moreover, in case of having entities that are also available in popular KGs such as DBpedia and Wikidata, they can be linked for enriching the information for the entities. For instance, in our KG the triples of Kostas Sloukas include his code ID in EuroLeague (i.e., 001926), and this information is also covered in his Wikidata page (`https://www.wikidata.org/wiki/Q2618541`).

```
You are given the classes and their path patterns of a Knowledge Graph
about basketball games.  With the go to, we indicate the class that you
need to visit for constructing a longer path.

#Path Patterns
*bball:Game
 ->bball:homeTeam->bball:Team (go to *bball:Team)
 ->bball:hasTeamBoxscore->bball:TeamBoxscore (go to *bball:TeamBoxscore)
 ->bball:hasAudience-> Literal (xsd:integer)
...
*bball:Team
 ->rdfs:label->Literal
 ->bball:teamCountry->bball:Country (go to *bball:Country)
...
*bball:TeamBoxscore
    ->bball:hasTeamStatline->bball:Statline (go to *bball:Statline)
...

#Extra Patterns
*The URIs include as suffixes identifiers and not names
*The persons names are given by the rdfs:label
with the pattern (Name Surname) in English
*The team names are given through rdfs:label in English

By using properties and classes from the above patterns,
please generate only a  SPARQL query  (without explanation)
for answering the question Q
```

Fig. 5: The prompt template for the Path and Literals Patterns Method

## 4 Experimental Evaluation

We first provide some statistics about the KG, comparative results for the QA over the KG by using ChatGPT and Deepseek and then we provide a discussion and specific research directions. The KG and the application is hosted in a single machine having an i5 core, 4 GB main memory and 50GB disk space.

### 4.1 Statistics of the KG

Table 1, provides some general statistics about the constructed KG, including the number of unique URIs, Literals, and others. The KG includes all the 25 seasons from 2000-2001 to 2024-25 and can be easily updated. As we can see, the KG covers over 5 million triples and it has a high number of triples with literals as objects, i.e., 84% of the total triples (they mainly describe the values of statistics). However, since the most literals refer to numerical values, they are repeated, thereby, the distinct literals are only 29 thousand.

Table 2 shows the number of URIs for each class, the number of triples where a URI of the class is either a subject or object and the average triples for the

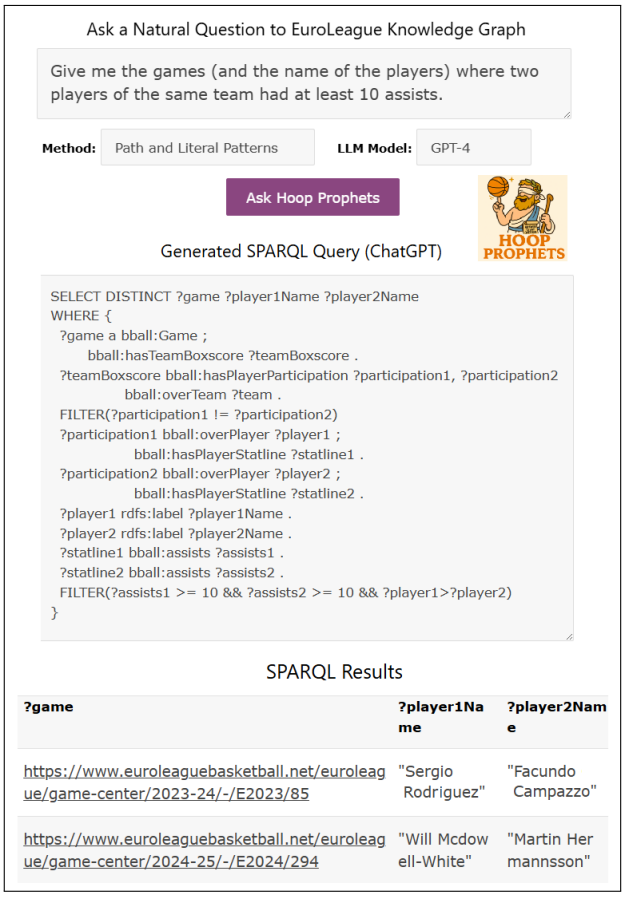


Fig. 6: Screenshots from the HoopProphets

URI of each class. As we can see, the KG includes more than 6 thousand games and 3 thousands players. For each game, there are on average over 25 triples per URI, and for each player over 55 triples. Moreover, the most URIs belong to Statline, i.e., over 140 thousands, whereas the highest number of average triples refer to teams, i.e., 435 triples per team.

### 4.2 The QA Benchmark over the KG and Experimental Setup

For the given ontology and KG, we have created a benchmark called *EuroLeague-QABench* having in total 100 competency questions. It includes 20 questions in five categories, according to the SPARQL clause needed for each query, i.e., SELECT, FILTER, ORDER BY, GROUP BY, and NEGATION. An example of each category in natural language, with the desired SPARQL query is given in Fig. 7. Moreover, for each natural question, we have asked the LLM to provide a paraphrased version including more informal (and usually less number of) words,

Table 1: General KG Statistics

| Type | Value |
|---|---|
| Number of Triples | 5,288,814 |
| Triples with Object URIs | 845,840 |
| Triples with Object Literals | 4,442,974 |
| Distinct URIs | 309,877 |
| Distinct Literals | 29,898 |
| Distinct Subjects | 307,750 |
| Distinct Objects | 333,573 |

Table 2: Statistics of the KG Classes

| Class | URIs | Triples | avg Triples/URI |
|---|---|---|---|
| Season | 25 | 6,351 | 254.0 |
| Game | 6,201 | 161,077 | 25.9 |
| Player | 3,234 | 180,443 | 55.7 |
| Team | 87 | 37,905 | 435.6 |
| Venue | 239 | 7,346 | 30.7 |
| Coach | 232 | 16,157 | 69.6 |
| Referee | 302 | 20,398 | 67.5 |
| Statline | 140,765 | 4,170,247 | 29.6 |

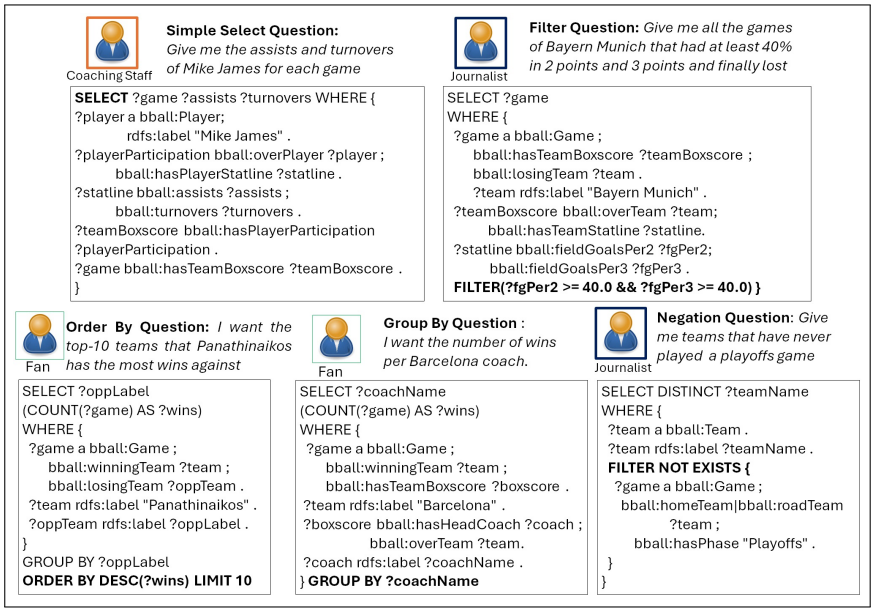


Fig. 7: Example with the questions of the benchmark

say $EuroleagueQABench_{par}$. For example, the question "Which teams missed out on the playoffs in 2023-24 although they won Real Madrid?" was paraphrased as "Who beat Real Madrid but still missed the 2023–24 Playoffs?".

Moreover, we also provide experiments for the same benchmark in Greek, i.e., $EuroleagueQABench_{gr}$ which was translated by native speakers, and in Chinese i.e., $EuroleagueQABench_{cn}$, which was translated automatically using ChatGPT. As a result, some questions in Chinese may not have been translated accurately.

**Experimental Setup.** We provide experiments for the above benchmark, by using the following LLMs, Deepseek [11] and ChatGPT [20], for versions 4 and 4-mini. We present comparative results for each LLM, by using i) the path and literals patterns method and ii) a baseline, in which we just give to the LLM a prompt with the list of all the classes and all the properties of the KG.

| Rank | Model | Method | SELECT | FILTER | ORDER | GROUP | NEG. | All Qs |
|---|---|---|---|---|---|---|---|---|
| 1 | GPT-4 | Patterns | **0.90** | **0.75** | **0.90** | 0.80 | 0.60 | **0.79** |
| 2 | DeepSeek | Patterns | 0.70 | 0.70 | 0.65 | **0.85** | **0.65** | 0.71 |
| 3 | GPT-4mini | Patterns | 0.55 | 0.15 | 0.50 | 0.30 | 0.30 | 0.36 |
| 4 | DeepSeek | Baseline | 0.40 | 0.25 | 0.40 | 0.25 | 0.20 | 0.30 |
| 5 | GPT-4 | Baseline | 0.45 | 0.10 | 0.25 | 0.25 | 0.20 | 0.25 |
| 6 | GPT-4mini | Baseline | 0.15 | 0.05 | 0.10 | 0.05 | 0.00 | 0.07 |

Table 3: Accuracy scores for the English version of the benchmark

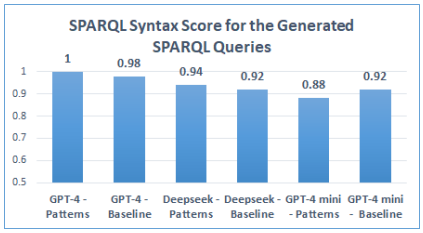


Fig. 8: SPARQL Syntax Score for the Generated SPARQL Queries

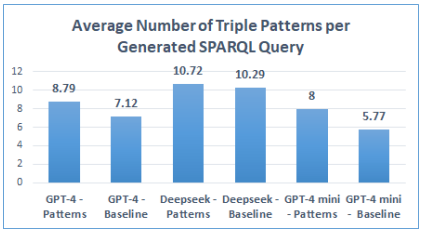


Fig. 9: Average Number of Triple Patterns per SPARQL Query

**Metrics.** We provide the results of three metrics: *Syntax Score*, i.e., the ratio of syntactically valid SPARQL queries (range $[0, 1]$), the *Accuracy*, i.e., the ratio of SPARQL queries that returned the desired result (range $[0, 1]$), and c) *Triple Patterns Number*, i.e., the number of Triple Patterns per SPARQL query.

**GitHub.** All the questions/versions of the benchmark, their gold standard, the generated SPARQL queries for each LLM and method, the results and the application code can be found in `https://github.com/mountanton/BasketballQA`.

### 4.3 Effectiveness for Natural QA using the LLMs

First, for the English benchmark, for each LLM we compare the results for the path and literals patterns and the baseline, and for the paraphrased and multilingual ones, we provide the results for the path and literals patterns method.

**Effectiveness Results on English Version.** As we can see in the last column of Table 3, the patterns method outperforms the baseline for all the LLMs, i.e., the accuracy for the baseline is less than 0.3 in all the cases. The LLM having the best results is GPT-4, with accuracy 0.79, whereas *Deepseek* also reaches high accuracy, i.e., 0.71. Regarding the query types, the results were better for SELECT, ORDER BY and GROUP BY queries, whereas the lowest accuracy obtained for the NEGATION queries. Fig. 8 shows that *Deepseek* is more vulnerable to syntax errors than *GPT-4*, whereas in Fig. 9, it is clearly that *Deepseek* constructs on average SPARQL queries with more (and usually redundant) triple patterns than the two *GPT* models.

**Common Errors.** In most cases, both *GPT-4* and *Deepseek* selected the correct patterns (which was not the case for *GPT-4 mini*). Concerning the most common errors, for the *GPT-4* were the following: i) it did not use the DISTINCT

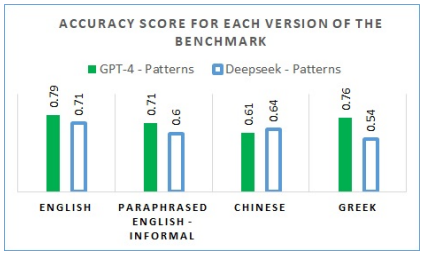


Fig. 10: Results on Multilingual and Paraphrased version

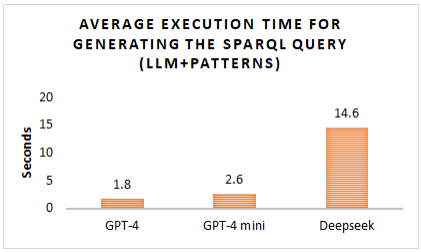


Fig. 11: Average Execution Time for each LLM with the patterns method

in some cases (so it returned duplicates), ii) wrong calculation of some statistics requiring average values, iii) in the negation, it used FILTER instead of FILTER NOT EXISTS. On the other hand for the *Deepseek* the main problems were: a) syntax errors, b) wrong calculation of some statistics requiring average values and c) extra filters and triple patterns that were either redundant or erroneous.

**Effectiveness Results on Paraphrased and Multilingual version.** Figure 10 shows the results for the other versions of the benchmark. First, for the paraphrased version, there is a decrease in the accuracy, approximately $-0.1$ for both models. Generally, the questions were more informal and contained fewer words, which resulted in many erroneous queries especially for the NEGATION questions. For the other categories the accuracy remained high. Concerning the multilingual versions, for the Greek version, GPT-4 had a high accuracy (i.e., 0.76) and in most cases, it correctly translated the Greek names, e.g., of the players, to their English name. On the other hand, for the *Deepseek* the accuracy for Greek was low (it failed in many cases to translate correctly the names), but it achieved higher accuracy for the Chinese version compared to *GPT-4*.

### 4.4 Efficiency and Monetary Cost for Natural QA using the LLMs

Fig. 11 shows the execution time for each LLM. *GPT-4* needs on average 1.8 seconds for generating the SPARQL query, i.e., it is approximately $8\times$ faster compared to *Deepseek*, which needs on average 14.6 seconds for each query. Concerning the monetary cost, *GPT-4* is $10\times$ more expensive, i.e., for 100 queries the price was 0.34$, whereas for *Deepseek* and *GPT-4 mini*, it was less than 0.03$.

### 4.5 Discussion and Limitations

As regards *GPT-4*, it generally produces correct queries, it has a high performance on the English and Greek questions, and it is quite fast. Some key limitations are that it forgets the distinct in some cases, and it can be expensive as the number of questions and/or patterns increases. Concerning *Deepseek*, it usually produces correct queries, it has high effectiveness for the English and Chinese version, and it is quite cheaper than GPT, i.e., more than $10\times$. On the other

hand, it adds more redundant triple patterns, it is more vulnerable in syntax errors and it is quite slower than the GPT models. Finally, the GPT-4mini is both fast and cheap, however, it produces much more erroneous SPARQL queries.

Regarding other limitations, the users tend to search with different names for the entities of a KG, e.g., they can use abbreviations for the teams (e.g., Barca instead of Barcelona) only the surnames of persons (Lessort instead of Mathias Lessort), and so forth. This is also clear for the results of the paraphrased version of the benchmark. One solution is to exploit autocomplete techniques, e.g., when a user types a surname, say Nunn, to send to the LLM the question including the full name of the player (e.g., Kendrick Nunn) or even the URI of the player.

## 5 Conclusion

In this paper, we proposed a pipeline including the steps of data collection, competency questions, ontology design, KG construction, and QA services using LLMs. We focused on the sports domain and specifically on basketball, and we used the steps of the pipeline to construct an ontology about basketball games and a KG for the EuroLeague Basketball. The KG includes 5 million triples for all the games (over 6 thousands games) from the season 2000-01 until 2024-25. Afterwards, we described an ontology paths and literals patterns method, and an application for generating SPARQL queries from natural questions (even multilingual ones). Regarding the experiments, we used a dedicated benchmark with 100 questions in three languages. For the english version we achieved accuracy score 0.79 using GPT-4 and 0.71 using Deepseek for the path and literals patterns method (instead of less than 0.3 for the baseline). For the multilingual, GPT-4 achieved for the Greek version accuracy 0.76 and for the Chinese 0.61.

As a future work, we plan to a) create a more interactive dialogue-based web application for enabling users to ask their own questions in any language, by using the proposed QA approach (such as [19]), b) extend the ontology for enriching the KG with more data, like the play-by-play of each game and more data for the rest entities, e.g., players, teams, by finding the same entities in external KGs (e.g., by using LODChain [17]), and c) create a service for automatically evaluating the results of SPARQL query generation over a benchmark.

## References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: ISWC. pp. 722–735. Springer (2007)
2. Avila, C.V.S., Vidal, V.M., Franco, W., Casanova, M.A.: Few-shot learning or rag in llm-based text-to-sparql? why not both?
3. Diallo, P.A.K.K., Reyd, S., Zouaq, A.: A comprehensive evaluation of neural sparql query generation from natural language questions. IEEE Access (2024)
4. Garijo, D.: Widoco: a wizard for documenting ontologies. In: ISWC Proceedings, Part II 16. pp. 94–102. Springer (2017)

5. Ghosh, I., Ramasamy Ramamurthy, S., Chakma, A., Roy, N.: Sports analytics review: Artificial intelligence applications, emerging technologies, and algorithmic perspective. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **13**(5), e1496 (2023). https://doi.org/10.1002/widm.1496
6. Hu, Y., Song, K., Cho, S., Wang, X., Foroosh, H., Yu, D., Liu, F.: Sportsmetrics: Blending text and numerical data to understand information fusion in llms. arXiv preprint arXiv:2402.10979 (2024)
7. Katris, C.: Exploring euroleague history using basic statistics. arXiv preprint arXiv:2301.02443 (2023). https://doi.org/10.48550/arXiv.2301.02443
8. Lampis, T., Ioannis, N., Vasilios, V., Stavrianna, D.: Predictions of european basketball match results with machine learning algorithms. Journal of Sports Analytics **9**(2), 171–190 (2023). https://doi.org/10.3233/JSA-220639
9. Li, A., Li, X., Zhu, B., Li, G.: Knowledge-graph-driven question-answer system for basketball. In: 5th International RICAI Conference. pp. 757–761. IEEE (2023)
10. Li, Y., Cao, J., Wang, Y.: Implementation of intelligent question answering system based on basketball knowledge graph. In: 4th IAEAC. pp. 2601–2604. IEEE (2019)
11. Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., et al.: Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437 (2024)
12. Lohmann, S., Link, V., Marbach, E., Negru, S.: Webvowl: Web-based visualization of ontologies. In: International Conference on Knowledge Engineering and Knowledge Management. pp. 154–158. Springer (2014)
13. Mandić, R., Jakovljević, S., Erčulj, F., Štrumbelj, E.: Trends in NBA and Euroleague basketball: Analysis and comparison of statistical data from 2000 to 2017. PloS one **14**(10), e0223524 (2019)
14. Mecharnia, T., d'Aquin, M.: Performance and limitations of fine-tuned llms in sparql query generation. In: GenAIK Workshop Proceedings. pp. 69–77 (2025)
15. Mountantonakis, M.: Services for Connecting and Integrating Big Numbers of Linked Datasets, vol. 50. IOS Press, Amsterdam, Netherlands (2021)
16. Mountantonakis, M.: Efficient statistical computation for k-player basketball lineups using semilattice structures. Electronics **14**(11) (2025)
17. Mountantonakis, M., Tzitzikas, Y.: Lodchain: Strengthen the connectivity of your rdf dataset to the rest lod cloud. In: ISWC. pp. 537–555. Springer (2022)
18. Mountantonakis, M., Tzitzikas, Y.: Generating sparql queries over cidoc-crm using a two-stage ontology path patterns method in llm prompts. ACM Journal on Computing and Cultural Heritage **18**(1), 1–20 (2025)
19. Mountantonakis, M., Tzitzikas, Y.: Tcrmq: An application for generating cidoc-crm sparql queries from text by using large language models (2025)
20. OpenAI: Chatgpt. https://chat.openai.com (June 2025)
21. Poveda-Villalón, M., Gómez-Pérez, A., Suárez-Figueroa, M.C.: OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation. International Journal on Semantic Web and Information Systems (IJSWIS) **10**(2), 7–34 (2014)
22. Sarlis, V., Papageorgiou, G., Tjortjis, C.: Leveraging sports analytics and association rule mining to uncover recovery and economic impacts in nba basketball. Data **9**(7), 83 (2024)
23. Singh, N.: Sport analytics: a review. learning **9**(11) (2020)
24. Strappazon, A., Granitzer, M., Egyed-Zsigmond, E., Mitrovic, J., Amor, M.B.: Instruct-to-sparql: A text-to-sparql dataset for training wikidata agents. In: ACM SIGIR Conference on Human Information Interaction And Retrieval. ACM (2025)
25. Taffa, T.A., Usbeck, R.: Leveraging llms in scholarly knowledge graph question answering. arXiv preprint arXiv:2311.09841 (2023)

26. Thabtah, F., Zhang, L., Abdelhamid, N.: NBA game result prediction using feature analysis and machine learning. Annals of Data Science **6**(1), 103–116 (2019)
27. Vinué, G., Epifanio, I.: Archetypoid analysis for sports analytics. Data Mining and Knowledge Discovery **31**, 1643–1677 (2017)
28. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM **57**(10), 78–85 (2014)
29. Wang, H., Sarker, A., Hosoi, A.: The effect of basketball analytics investment on national basketball association (nba) team performance. JSE, SAGE (2025)
30. Weikum, G.: Knowledge graphs 2021: A data odyssey. Proceedings of the VLDB Endowment **14**(12), 3233–3238 (2021)
31. Xia, H., Yang, Z., Wang, Y., Tracy, R., et al.: Sportqa: A benchmark for sports understanding in large language models. arXiv preprint arXiv:2402.15862 (2024)
32. Yang, S., Teng, M., Dong, X., Bo, F.: Llm-based sparql generation with selected schema from large scale knowledge base. In: China Conference on Knowledge Graph and Semantic Computing. pp. 304–316. Springer (2023)