Clover Quiz: a trivia game powered by DBpedia

Guillermo Vega-Gorgojo [a,b]
[a] *Don Naipe, Oslo, Norway*
[b] *Department of Informatics, University of Oslo, Norway*
E-mail: guiveg@ifi.uio.no

Abstract. DBpedia is a large-scale and multilingual knowledge base generated by extracting structured data from Wikipedia. There have been several attempts to use DBpedia to generate questions for trivia games, but these initiatives have not succeeded to produce large, varied, and entertaining question sets. Moreover, latency is too high for an interactive game if questions are created by submitting live queries to the public DBpedia endpoint. These limitations are addressed in Clover Quiz, a turn-based multiplayer trivia game for Android devices with more than 200K multiple choice questions (in English and Spanish) about different domains generated out of DBpedia. Questions are created off-line through a data extraction pipeline and a versatile template-based mechanism. A back-end server manages the question set and the associated images, while a mobile app has been developed and released in Google Play. The game is available free of charge and has been downloaded by more than 5K users since the game was released in March 2017. Players have answered more than 614K questions and the overall rating of the game is 4.3 out of 5.0. Therefore, Clover Quiz demonstrates the advantages of semantic technologies for collecting data and automating the generation of multiple choice questions in a scalable way.

Keywords: knowledge extraction, DBpedia, trivia game, multiple choice question, mobile app

1. Introduction

Wikipedia is the most widely used encyclopedia and the result of a truly collaborative content edition process.[1] There are 295 editions of Wikipedia corresponding to different languages, although the English Wikipedia is the largest with more than 5.4 million entries. Articles do not only include free text, but also multimedia content and different types of structured data like so-called infoboxes and category declarations. Wikipedia has an impressive breadth of topical coverage that includes persons, places, organisations, and creative works. The social and cultural impact of Wikipedia is quite significant: it is the fifth most popular website according to Alexa,[2] while Wikipedia's content is extensively used in education, journalism, and even court cases.[3] Despite the vastness and richness of Wikipedia, its content is only accessible through browsing and free-text searching. To overcome this limitation, the DBpedia project builds a knowledge base by extracting structured data from the different Wikipedias [12]. As a result, the latest release of the English DBpedia (2016-10) describes 6.6 million entities and contains 1.7 billion triples [16].

DBpedia constitutes the main hub of the Semantic Web [8, ch. 3] and is employed for many purposes such as entity disambiguation in natural language processing [11]. An appealing application case is the generation of questions for trivia games from DBpedia. Some preliminary attempts can be found in the literature [4, 9, 13, 14, 18], but these initiatives have fallen short due to simple

[1]https://www.wikipedia.org/
[2]http://www.alexa.com/topsites
[3]https://en.wikipedia.org/wiki/Wikipedia#Cultural_impact

question generation schemes that are not able to produce varied, large, and entertaining questions. Specifically, supported question types are rather limited, reported sizes of the generated question sets are relative low (in the range of thousands), and no user base seems to exist. Moreover, some of these works create the questions by submitting live queries to the public DBpedia endpoint, hence latency is too high for an interactive trivia game, as reported in [13].

The hypothesis is that creating questions from DBpedia can be significantly improved by splitting this process in a data extraction and a versatile question generation stages. This approach can produce varied, numerous, and high-quality questions by declaratively specifying the classes and question templates of the domains of interest. The generated questions can then be hosted in a back-end server that meets the latency requirements of an interactive trivia game. The target case is Clover Quiz, a turn-based multiplayer trivia game for Android devices in which two players compete over a clover-shaped board by answering multiple choice questions from different domains. This paper presents the outcomes of this project, including the mobile app and actual usage information of the players that have downloaded the game through Google Play.

The rest of the paper is organized as follows: Section 2 presents the game concept of Clover Quiz. Section 3 describes the data extraction pipeline, while Section 4 explains the question generation process. The design of the back-end server and the mobile app is addressed in Section 5. Section 6 deals with the actual usage of Clover Quiz, including user feedback and latency measures. Next, Section 7 draws some lessons learned. The paper ends with a discussion and future work lines in Section 8.

2. Game concept

Clover Quiz is conceived as a turn-based multiplayer trivia game for Android devices. In an online match, two players compete over a clover-shaped board. Each player has to obtain the 8 wedges in the board by answering questions on different domains. The player with the floor can choose any of the remaining wedges and then respond to a question on the corresponding domain. If the answer is correct, the player gets the wedge and can continue playing, but if it is incorrect, the floor goes to the opponent. Once a player obtains the 8 wedges, there is a duel in which each player has to answer the same 5 questions in a row. The match is over if the player with the 8 wedges wins the duel. In other case, this player loses all the wedges and the match continues until there is a duel winner with the 8 wedges.

The target audience of Clover Quiz corresponds to casual game players with an Android phone, in the age range of 18-54, high school/university level education, and Spanish- or English-speaking. Importantly, target users are not supposed to know anything about the Semantic Web and do not require a background on Computer Science or Information Technology. Since the game is purposed for mobile devices, user typing should be limited as much as possible. For this reason, Clover Quiz employs multiple choice questions with 4 options; note that this is also the solution adopted by other mobile trivia games like QuizUp[4] and Trivia Crack.[5]

Clover Quiz includes questions from the following domains: Animals, Arts, Books, Cinema, Geography, Music, and Technology – all of them have a good coverage in DBpedia [12] and are arguably of interest to the general public. About the generation of questions, an important design decision is whether to prepare the questions beforehand or to submit live queries to DBpedia. The latter option was discarded due to the complexity of the question generation process and to the stringent requirements of interactive applications (like Clover Quiz) that cannot be met by the public SPARQL endpoint over the DBpedia dataset – queries to the public DBpedia endpoint can easily take several seconds and periods of unavailability are relatively common, according to the tests carried out in the inception phase of the game. Instead, the question set of Clover Quiz is generated in advance and deployed in a back-end server. This architecture corresponds to the crawling pattern employed in some Semantic Web applications [8, ch. 6].

[4]https://play.google.com/store/apps/details?id=com.quizup.core

[5]https://play.google.com/store/apps/details?id=com.etermax.preguntados.lite

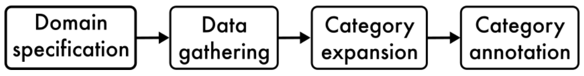


Figure 1. Overview of the data extraction process.

3. Data extraction

The goal of the extraction phase is to gather the data of interest from DBpedia and produce a consolidated dataset that can be easily exploited to generate multiple choice questions. This is accomplished through a series of steps that are graphically depicted in Figure 1. This workflow is supported with a collection of scripts coded in Javascript, while all generated input and output files are in JSON format [5]. In the first stage, a *Domain specification* file is authored with the instructions for retrieving data from a target endpoint – in this case, the English DBpedia public endpoint.[6] This specification file identifies the classes in the domain of interest, e.g. `Museum`, `Painting`, or `Painter` in Arts. Each class is associated with a SPARQL query for retrieving the corresponding DBpedia entities. Note that the SPARQL query is trivial if there is already a class in the DBpedia ontology that perfectly matches the intended concept, e.g. `dbo:Museum`. However, a suitable DBpedia class is not always available or target entities are members of other classes, thus requiring to craft complex queries like the one in Listing 1.

Listing 1: SPARQL query for retrieving the entities of the `Painting` class

```
select distinct ?X where {
?X dct:subject ?S .
?S skos:broader{,4} dbc:Paintings .
{ {?X a dbo:Artwork .}
  UNION {?X (dbo:author | dbp:author) [] . }
  UNION {?X (dbo:artist | dbp:artist) [] . }
  UNION {?X (dbo:museum | dbp:museum) [] . }
  UNION {[] dbp:works ?X .} } . }
```

A domain specification file also identifies the literals to be extracted for the entities of a target class – like labels, years, or image URLs – by providing the corresponding datatype properties used in DBpedia. In addition, relations between entities of different classes are also defined; simple cases just involve an object property, e.g. `dct:subject` for getting the Wikipedia categories of `Painting`. Unfortunately, the structure of DBpedia is not very regular and it is common to find alternative properties with similar meaning. As a result, more complicated queries are frequently needed to extract relations between DBpedia entities – see for example the query in Listing 2.

Listing 2: SPARQL query for retrieving the cities where museums are located

```
select distinct ?entA ?entB where {
?entA a dbo:Museum .
?entB a dbo:Settlement .
?entA (dbo:location | dbp:location |
       dbo:city | dbp:city){1,3} ?entB . }
```

In the *Data gathering* stage, a script takes a specification file as input and systematically queries DBpedia to retrieve the data available of the target domain. Essentially, the script gathers the entities belonging to each class, their literals, and their relations with other entities, as defined in the domain specification file. For every DBpedia entity found, the script also obtains the number of triples with that individual as subject (`outlinks`) and the number of triples with that individual as object (`inlinks`) – these measures are employed to estimate the popularity of an individual in the question generation phase (see Section 4). All queries are paginated using the *LIMIT* and *OFFSET* SPARQL keywords, while a parameter restricts the number of concurrent queries sent to the endpoint. Importantly, the script runs in an incremental way, saving the work in case of errors such as a temporal unavailability of DBpedia. The output of this stage is a file with a JSON object for every entity found, e.g. "The Surrender of Breda" in Listing 3.

The snippet in Listing 3 includes the URI of the entity, the collected literal values, the `outlinks` and `inlinks`, and the relations to entities from other classes. The last item, `painting_categories`, corresponds to the list of Wikipedia categories that the source Wikipedia article is endowed. Wikipedia contributors annotate articles with suitable categories that are organized into a hierarchy that reflects the notion of "being a subcategory of" [3]. Wikipedia categories represent a precious knowledge resource that can be extremely powerful for adding variety and uniqueness to the question set in Clover Quiz. In this running ex-

[6]http://dbpedia.org/sparql

Listing 3: JSON object generated in the data gathering stage for a sample entity

```
{ "uri": "http://dbpedia.org/resource/The_Surrender_of_Breda",
"label": [ {"es": "La rendición de Breda"}, {"en": "The Surrender of Breda"} ],
"image": "http://en.wikipedia.org/wiki/Special:FilePath/Velazquez-The_Surrender_of_Breda.jpg",
"outlinks": "85",
"inlinks": "2",
"painter": [ "http://dbpedia.org/resource/Diego_Velázquez" ],
"museum": [ "http://dbpedia.org/resource/Museo_del_Prado" ],
"painting_categories": [ "http://dbpedia.org/resource/Category:1634_paintings",
    "http://dbpedia.org/resource/Category:Velazquez_paintings_in_the_Museo_del_Prado",
    "http://dbpedia.org/resource/Category:War_paintings",
    "http://dbpedia.org/resource/Category:Horses_in_art" ] }
```

Table 1
Summary of the data extraction process for the different domains

	Animals	**Arts**	**Books**	**Cinema**	**Geo**	**Music**	**Tech**	**TOTAL**
# of classes	6	22	9	10	19	17	18	101
# of entities	82,874	223,022	141,621	353,361	251,927	349,443	162,941	1,565,189
Category data (MB)	61	232	149	262	168	231	59	1,162
Annotated data (MB)	67	108	79	236	122	202	115	929

ample, the category hierarchy can be exploited to gather the year where "The Surrender of Breda" was completed and other relevant facts such as being a Spanish painting, with animals, and from the Baroque period that are derived from broader categories. Since there are more than one million Wikipedia categories, the aim of the *Category expansion* stage is to gather a relevant subset of the category hierarchy for the domain of interest. A script automatically generates this subset by extracting all the categories found in the data gathering stage and then querying DBpedia to obtain the set of broader categories for each specific one (using the property `skos:broader`). The script continues recursively with each new category found until navigating 4 levels up in the hierarchy – this parameter was chosen because it gives a good domain coverage with a reasonable size.

The last stage of the data extraction process involves the authoring of a *Category annotation* file in which a set of Wikipedia categories of interest are specified for the target classes, e.g. `dbc:Baroque_paintings` for the `Painting` class. A script takes as input this file, as well as the extracted data and the expanded category files generated in the previous stages. For every candidate category, the script obtains the expanded set of subcategories and then evaluates the individuals of the target class. In the example above, "The Surrender of Breda" is annotated as `Baroque_paintings` because `dbc:Velazquez_paintings_in_the_Museo_del_Prado` is a subcategory of `dbc:Baroque_paintings`. The obtained annotations are added to the corresponding JSON object, e.g. Listing 4 shows the annotations of the running example.

Listing 4: Annotations added to the JSON object in Listing 3 after the data annotation stage

```
"year": 1634,
"Baroque_paintings": true,
"Spanish_paintings": true,
"War_paintings": true,
"Animals_in_art": true
```

Table 1 gives some figures about the number of classes specified, the number of entities extracted from DBpedia, the size of the expanded category files, and the size of the annotated data files for each domain in Clover Quiz.

4. Question generation

A multiple choice question consists of a *stem* (the question), a *key* (the correct answer), and *distractors* (a set of incorrect, yet plausible, answers) [1]. In Clover Quiz, the challenge is to produce numerous, varied, and entertaining questions in a scalable way. Moreover, a question difficulty estimator is required to match the questions to the players' skills during the game – intuitively, novice players should get easy questions, while experienced players should get more challenging questions as they progress through the game. To comply with these requirements, a template-based question generator is devised. It consists of a script that takes as input a list of question templates and an annotated data file of a domain, as produced at the end on the data extraction pipeline (see Section 3).

The question generator supports different template types in order to allow the creation of varied questions, namely: `image`, e.g. *Which is the painting of the image?*; `boolean`, e.g. *Which is the modernist building?*; `boolean negative`, e.g. *Which is NOT an Ancient Greek sculptor?*; `group`, e.g. *Which is the artistic style of the painter {{painter.label}}?* (options: *Gothic, Renaissance, Baroque, Mannerist, Romantic*); `date`, e.g. *When was {{painter.label}} born?*; `greatest`, e.g. *Which country has the largest population?*; `numeric`, e.g. *Which is the population of {{city.label}}?*; `relation`, e.g. *Who is the painter of "{{painting.label}}"?*; and `relation negative`, e.g. *Which castle is NOT in {{country.label}}?* – the latter two template types connect entities from two classes, while the others just involve a single class.

A question template is just a JSON object with a set of key-values, e.g. Listing 5. This template can be used to generate questions in Spanish or English – the target languages of Clover Quiz – by choosing the appropriate value of the key *question*. The core part of a template is the key *class* that defines the entities in the annotated data file to which the template applies; in Listing 5, target entities are members of the `Painting` class and have to include the following JSON keys: `image`, `Baroque_paintings`, and `Animals_in_art`. A template can also specify a *min_score* to filter out those candidates with a lower popularity score – this is computed with this formula: $pop\_score = outlinks + 10 * inlinks$.[7] Inspired by the PageRank algorithm [15], the rationale of the employed popularity score is to differentiate well-known entities from obscure ones. Concerning the rest of the items in the template, *image_prop* identifies the JSON key with the image URL, *topic* is employed for classification purposes, and *dif_level* is a subjective rating of the difficulty of the questions generated with a template – ranging from 0 (very easy) to 10 (very difficult).

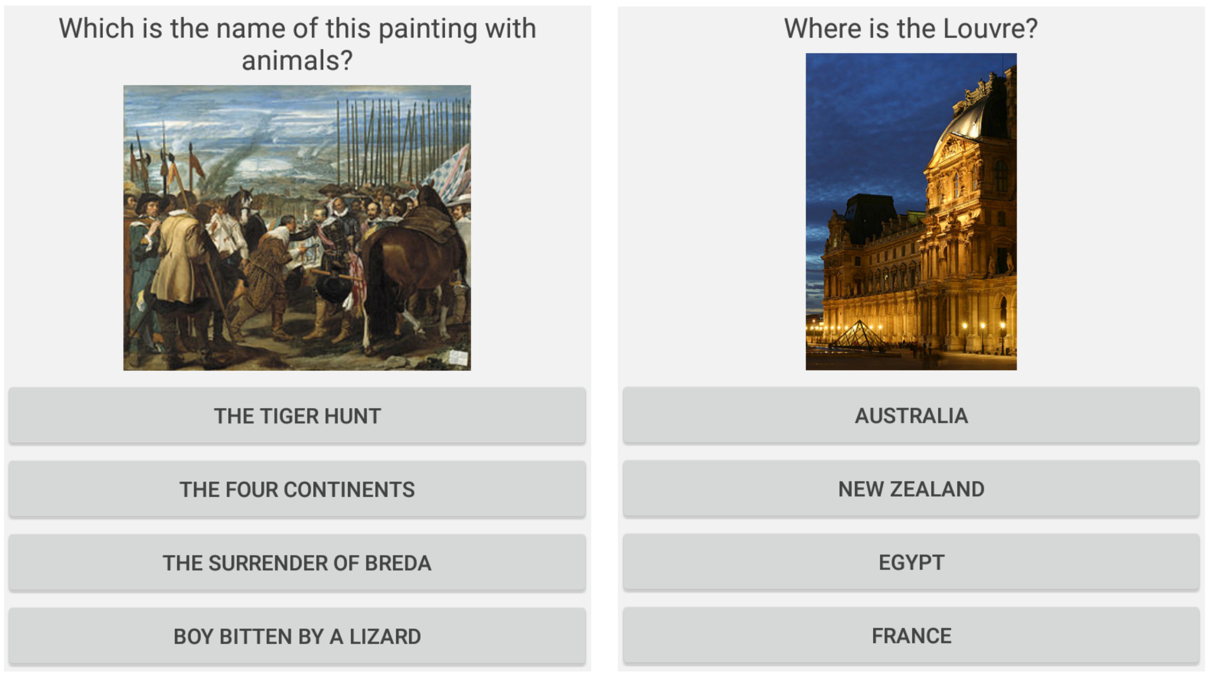
Listing 5: Example of an `image` single class question template

```
{ "question": [ {"en": "Which is the name of this painting
    with animals?"},
    {"es": "¿Cuál es el nombre del cuadro con animales
    de la imagen?"} ],
"class": "Painting.image.Baroque_paintings.Animals_in_art",
"min_score": 50,
"image_prop": "image",
"topic": ["baroque"],
"dif_level": 1 }
```

When the template in Listing 5 is evaluated, the question generator first obtains the set of paintings that comply with the requirements, e.g. "The Surrender of Breda". It will then generate a question for each occurrence by getting the image URL (to support the question) and the label of the painting (this will be the correct answer). Finally, the script will prepare three lists of distractors that correspond to distinct difficulty levels. This is performed by taking a random sample of 50 paintings in the same set, estimating the similarity of each element to the correct answer, discarding the less similar distractors, and finally preparing the three lists. Note that a question is more difficult if the distractors are closer to the correct answer [2], so similarity is computed with a measure based on Jaccard's coefficient [10] that is defined for every class by providing the array keys, e.g. `painting_categories`, boolean keys, e.g. `Baroque_paintings`, and date-based keys, e.g. `year`, of the target entities. This way, Figure 2(left) shows the question created with the template above when applied to "The Surrender of Breda" painting. Variations of this template can be created very easily, e.g. switching from Baroque to Renaissance paintings, or from animal to still life paintings.

All template types have a similar structure, although double class templates are slightly different. Listing 6 shows the template employed to generate the question in Figure 2(right). This tem-

[7] `outlinks` and `inlinks` were obtained in the data extraction pipeline (see Section 3). Note that `outlinks` aggregates literal triples and outgoing RDF links, while `inlinks` only counts incoming RDF links; `inlinks` is multiplied by a factor of 10 in *pop_score* to stress its importance.


Which is the name of this painting with animals?

Where is the Louvre?

Figure 2. Sample questions from the Arts domain obtained with the mobile app of Clover Quiz. The distractors correspond to the “easy list” – this is especially evident in the second example that includes countries quite dissimilar to France, i.e. non-members of the EU, in different continents, non-French speaking, and so on.

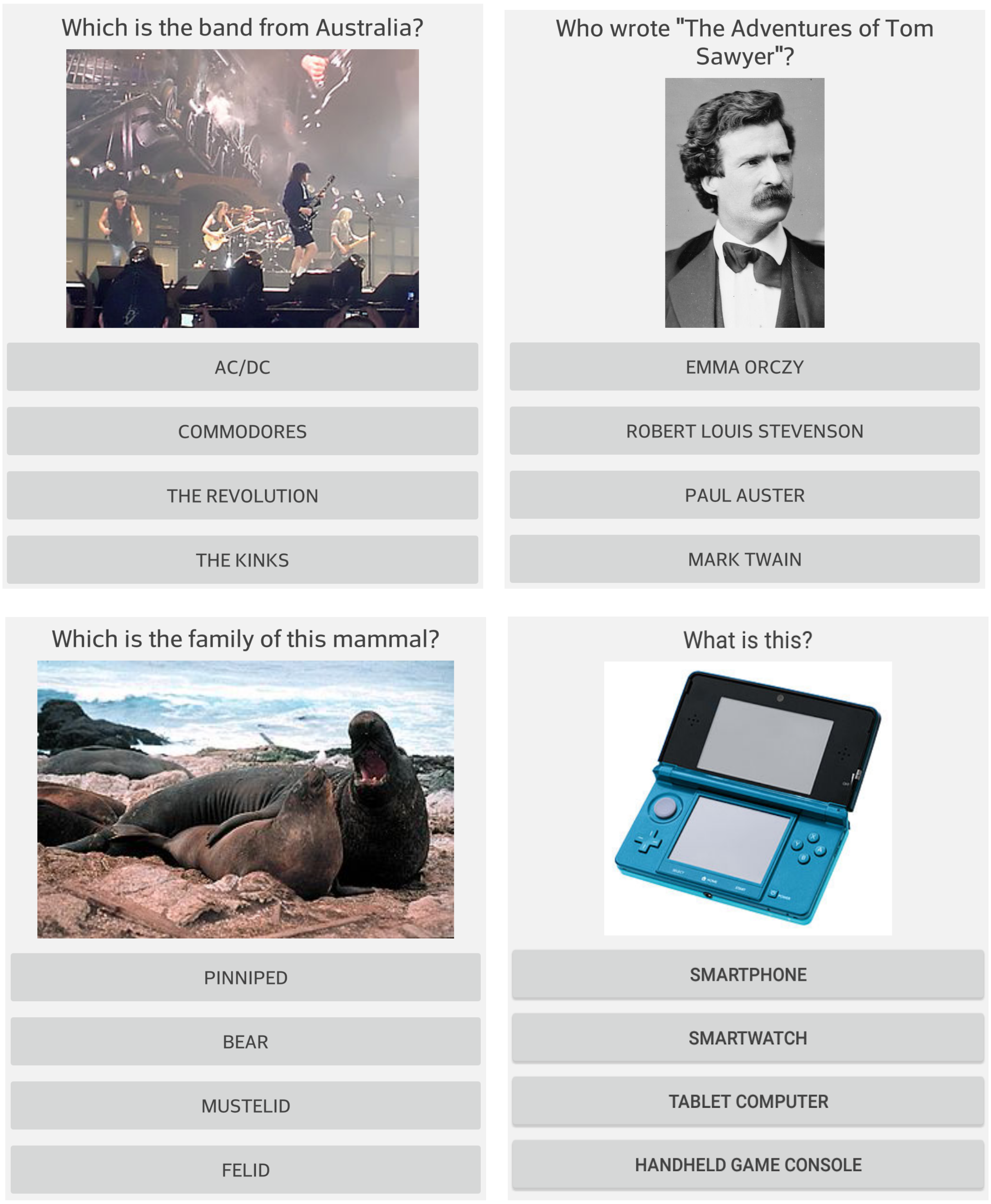
plate involves two classes (`Museum` and `Country`) connected through the property `country`. This property is functional, since a `Museum` can only be located in one `Country` – entities that do not comply with this restriction are silently filtered out by the question generator. Note that the template is not inverse because answers are members of *classB*, i.e. `Country` in this case. The rest of elements in the template are similar to the ones in Listing 5. Again, it is very easy to prepare variations of this template, e.g. reversing the question in order to choose the country of a particular museum (this new template will ask for members of *classA*, and will thus be inverse). Figure 3 shows additional questions from non-Arts domains.

Listing 6: Example of a `relation` double class question template

```
{ "question": [ {"en": "Where is the {{classA.label}}?"},
    {"es": "¿Dónde está el {{classA.label}}?"} ],
"classA": "Museum.image",
"classB": "Country.Member_states_of_the_United_Nations",
"prop": "country",
"inverse": false,
"functional": true,
"min_scoreA": 300,
"min_scoreB": 300,
"image_propA": "image",
"topic": ["museums"],
"dif_level": 0 }
```

After creating the questions associated to a template, the script computes an estimator of the questions’ difficulty. It relies on the popularity of the involved entities (see *pop_score* above) and the template difficulty level assessment (see *dif_level* above). The former is employed to assign a within-template difficulty score, while the latter provides a between-template difficulty correction, e.g. a question about the completion year of a painting is arguably more difficult than asking the name of the same painting, so the *dif_level* of the latter template should be higher. With the computed difficulty estimator, questions are then sorted and unique identifiers are given to facilitate their retrieval during the game.

Table 2 presents some aggregated figures of the question set generated for Clover Quiz. The overall process consisted on the creation of several “meta-templates” for every domain (20 to 50, typically) and then preparing the specific templates, e.g. Listings 5 and 6. The rationale is to produce more cohesive questions related to specific topics (like Romanesque, Gothic, Renaissance, Baroque,



Which is the band from Australia?

AC/DC

COMMODORES

THE REVOLUTION

THE KINKS

Who wrote "The Adventures of Tom Sawyer"?

EMMA ORCZY

ROBERT LOUIS STEVENSON

PAUL AUSTER

MARK TWAIN

Which is the family of this mammal?

PINNIPED

BEAR

MUSTELID

FELID

What is this?

SMARTPHONE

SMARTWATCH

TABLET COMPUTER

HANDHELD GAME CONSOLE

Figure 3. Sample questions from the Music, Books, Animals and Technology domains obtained with the mobile app of Clover Quiz.

Table 2
Summary of the question generation process for the different domains. There are significantly more English questions in Arts and Books because Spanish labels were missing in many DBpedia entities in these domains

	Animals	**Arts**	**Books**	**Cinema**	**Geo**	**Music**	**Tech**	**TOTAL**
# of templates	125	269	387	295	724	767	374	2,941
# of questions (Spanish)	15,342	18,121	23,580	49,208	24,086	36,136	21,014	187,487
# of questions (English)	15,347	27,523	46,403	50,199	24,484	36,075	21,017	221,048

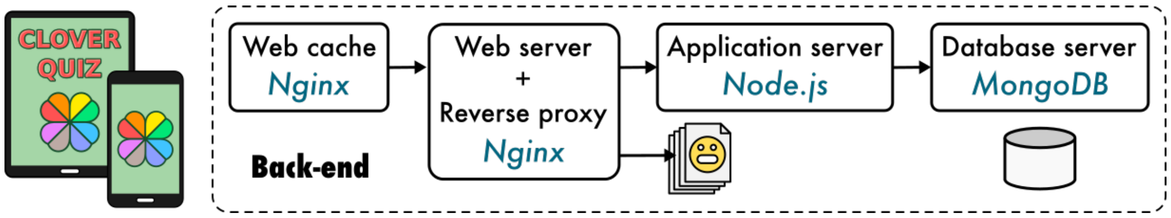


Figure 4. System architecture of Clover Quiz.

etc. in Arts) by partitioning the space in smaller and more coherent sets. The downside is that more templates are needed, although the required effort was kept low due to the massive use of copy&paste from the “meta-templates”.

5. Back-end sever and mobile app

After generating the question set of Clover Quiz, the next step is the system design. Figure 4 outlines the overall architecture, split into the mobile app and the back-end server. This separation is purposed to keep the mobile app as lightweight as possible, while the server is in charge of delivering the questions and associated images – note that questions in Clover Quiz are supported with more than 37K low-resolution images, totalling 1.12 GB. To simplify the back-end, a key design decision was to embrace the JSON format to avoid data transformations of the question set, already in JSON. Due to this, a MongoDB database is employed – MongoDB is a scalable and efficient document-based NOSQL system that natively uses JSON for storage.[8]

The role of the application server is to handle question requests without exposing the database server to the mobile app directly. In this way, the security of the database is not compromised and an eventual upgrade or replacement of the database component does not require changes in the mobile app. Since JSON was derived from JavaScript and is commonly employed with this language, a natural decision was to use Node.js for the application server. Node.js is a popular JavavaScript runtime environment for executing server-side code.[9]

Application servers are purposed for handling dynamic content, but they are not very strong for serving static content. Since 67% of the questions in Clover Quiz have an associated image, fast static file serving is an important requirement. This is addressed through the use of Nginx,[10] an efficient and fast performant Web server that shines at serving static content and can also be used as a reverse proxy [17]. Thus, Nginx was configured to host the game images and to forward question requests to the application server. In addition, another Nginx box was set up as a Web cache to improve performance and reduce the back-end load.

Regarding the mobile app, an Android version of the game described in Section 2 was coded. It can be played in phones and tablets and the user interface is built following the Material Design guidelines[11] – see sample snapshots in Figure 5. An essential functionality is the matchmaking of players that was implemented using Google Play Games

[8]https://www.mongodb.com/

[9]https://nodejs.org/
[10]https://nginx.org/
[11]https://material.io/

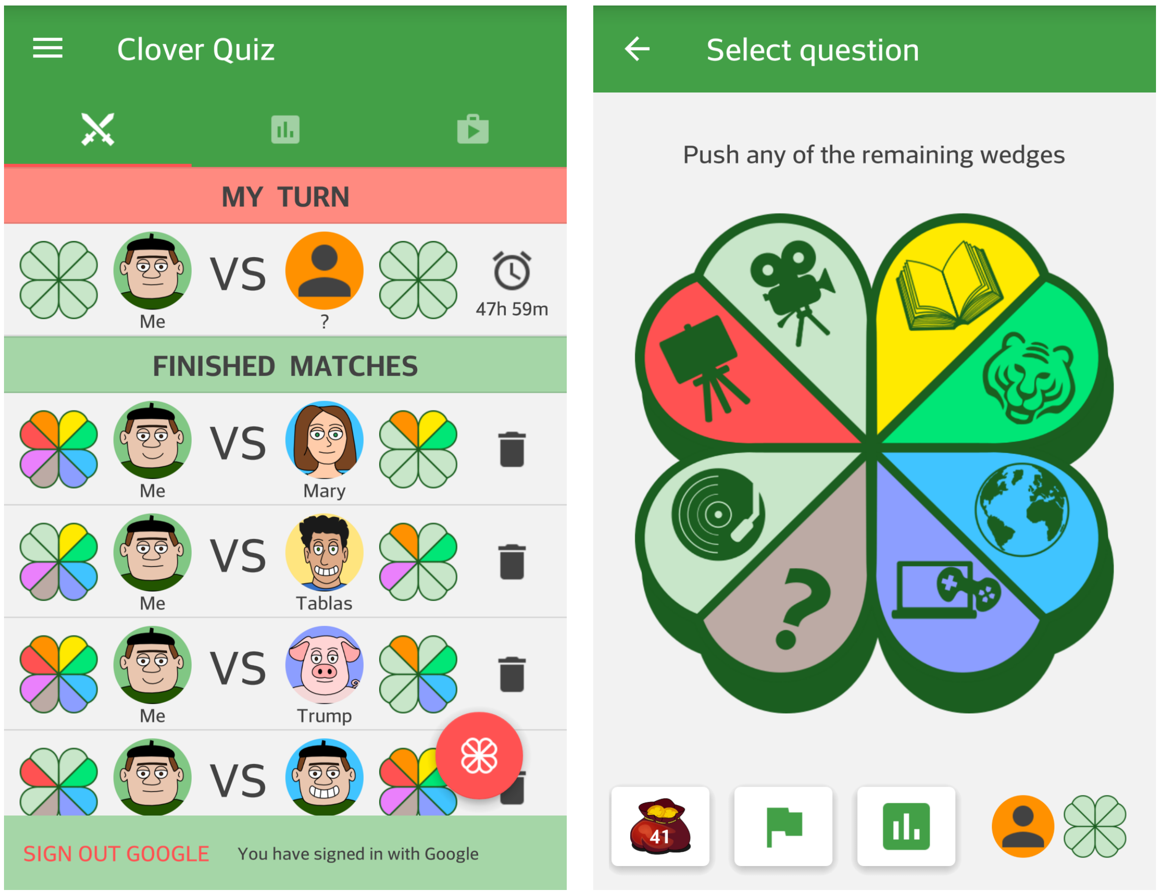


Figure 5. Sample snapshots of the mobile app of Clover Quiz.

Services[12] that provides a convenient and simple API for turn-based multiplayer games. This way, it is possible to initiate a match against a random player or to invite a friend. The initial screen of the game presents a list of pending invitations, ongoing and finished matches – see Figure 5(left) for an example.

After selecting a match, a clover-shaped board is displayed with 8 wedges corresponding to the different domains – see Figure 5(right). The player can push any of the available wedges, e.g. `Music`, then select a subtopic, e.g. `Heavy metal`, and finally answer the question posed. The mobile app keeps a player profile that is used as a basis to select a suitable question; specifically, there are 6 different expertise levels for each subtopic that controls the difficulty of the questions, along with a randomisation effect. In addition, the player profile keeps track of the last 5,000 questions posed to avoid repetitions. Player profiles are saved in the cloud through Google Play Games Services, thus allowing to save players' progression and continue from any device. It is also worth mentioning that the mobile app includes additional features such as a single-player mode, statistics, leaderboards and achievements. Players can also report a problem in a question by simply pushing a "Report problem" button that is always included in the question result screen.

[12]https://developers.google.com/games/services/

6. Clover Quiz in practice

The game was released for Android devices on March 11, 2017 under the names 'Clover Quiz' in English and 'Trebial' in Spanish. It is available

Table 3
Overview of the questions answered from March 11 to August 1

Set	Questions answered	Correct responses	Wrong responses	Problem reports
Spanish	597,771 (100.0%)	382,431 (64.0%)	215,340 (36.0%)	1,915 (0.3%)
English	16,899 (100.0%)	10,106 (59.8%)	6,793 (40.2%)	20 (0.1%)

free of charge through Google Play[13] and is part of the catalogue of Don Naipe,[14] a sole proprietorship company specialized in Spanish card games for mobile devices. Clover Quiz was promoted with an in-house ad campaign that ran from March 13 to March 16, i.e. other Android games by Don Naipe[15] showed interstitial ads about Clover Quiz.

At the time of this writing (August 2017), more than 5K users have downloaded the game. Table 3 shows some statistics of the questions answered during this period. It can look striking that most of the requested questions were in Spanish, but this basically reflects the user base of Don Naipe (note that Clover Quiz has been only promoted with in-house ads). Approximately two thirds of the questions were correctly answered, while each player has taken 122 questions on average, thus indicating a reasonable engagement with the game. Interestingly, the number of problem reports is quite low and concentrated in a small set of questions. A subsequent audit served to spot some problems: a group type template with a wrong option, several animals with misleading images, and one intriguing case in which `Body louse` was classified as a `Primate` – the reason is that this parasite is annotated in Wikipedia with the category `Parasites of humans` that is a subcategory of `Humans`.

Clover Quiz users have also given feedback through Google Play. Specifically, the average rating is 4.3 out of 5.0 and users' comments are generally very supportive: there are some suggestions of new domain areas, e.g. sports, and also a complain about a server failure on April 14 – there was a system reboot, and the question back-end was not automatically restarted, now it is up again.

The latency of the production back-end server was evaluated in May 2017. *curl*[16] was employed to measure the total response time of 1000 random questions. The client machine ran the experiment in Oslo, while the back-end is deployed in Amsterdam. The average response time was $0.107s$ with a standard deviation of $0.047s$. Similarly, 1000 random images hosted in the back-end were requested with *curl*, taking $0.127s$ on average with a standard deviation of $0.041s$. The reported latencies are quite low and perfectly acceptable for an interactive trivia game. Indeed, Clover Quiz users have not yet complained about performance.

7. Lessons learned

On the architecture of Semantic Web applications, a lesson learned is **the use of the crawling approach for consuming DBpedia data and its transformation to JSON to facilitate data processing**. If this process can be done off-line, – as in the case of Clover Quiz for the generation of the question set – consumer applications can be kept simple and with low latency. The downside is the replication of data and that applications may work with stale data. Regarding DBpedia, there is already a lag with Wikipedia, since the generation of DBpedia is dump-based with a typical periodicity of 1–2 releases per year [12]. Thus, data freshness can be ensured by re-crawling DBpedia after a new release is available.

DBpedia is an amazingly comprehensive and vast structured dataset, but DBpedia is also messy: there are multiple properties with essentially the same meaning, e.g. `dbp:birthPlace` and `dbp:placeOfBirth`; entities are not always members of the right classes, for example, `dbr:Beyoncé` is not a member of `dbo:MusicalArtist`; classes may be broader than expected, e.g. most of the entities in `dbo:Country` correspond to former countries and empires. As a result, **consuming DBpedia data requires a thorough examination of the target domains** – indeed, Section 3 gives several examples of complicated queries in the data extraction process of Clover Quiz because of this

[13]https://play.google.com/store/apps/details?id=donnaipe.trebial
[14]http://donnaipe.com/
[15]https://play.google.com/store/apps/developer?id=Don+Naipe
[16]https://curl.haxx.se/

messiness of DBpedia. In addition, some curation of the extracted data may also be needed.

A substantial part of the riches of DBpedia correspond to Wikipedia categories. **Wikipedia editors have invested a tremendous effort on the annotation of categories that can be exploited with DBpedia, although special care should be taken to avoid pitfalls**. More specifically, Wikipedia categories are a kind of "folksonomy", so problems can arise if they are handled as a strict class taxonomy – see the issue with `Body louse` in Section 6. In this respect, the category annotation script employed in Clover Quiz can be configured to limit the number of category levels considered in order to restrict undesired consequences of the category hierarchy. Furthermore, **DBpedia users should be aware that categorization of entities is unequal, i.e. an entity may not be included in a category although it should**. This is quite challenging for the generation of questions in Clover Quiz, e.g. a question about Baroque paintings would be incorrect if a Baroque painting was wrongly included as a distractor due to an incomplete categorization. To circumvent this problem, the target sets in Clover Quiz templates are carefully defined to limit the impact of missing information; in the previous example, the corresponding template defines a target set comprised of paintings from the Gothic, Renaissance, Baroque and Romantic movements, hence any non-categorized painting is silently discarded.

Overall, **the template-based mechanism employed to generate the question set and the popularity score used to rank the difficulty of questions have worked very well**. Although the number of templates defined in Clover Quiz is not small (see Table 2), this is mainly due to the creation of multiple template variations. In this way, the class space is partitioned in smaller and more coherent sets, e.g. the template employed to generate question Figure 2(left) is replicated for other movements like Modern Art or Impressionism. About the employed popularity score, it is a cheap measure for estimating the difficulty of questions that generally works very well, e.g. The Beatles is the most popular band and United States the most popular country. However, this estimator reproduces similar bias as Wikipedia,[17] for example, the Ecce Homo at Borja[18] is an unremarkable painting that became an Internet phenomenon due to a failed restoration attempt – this is the most popular Spanish painting according to the employed popularity score.

8. Discussion

There are several works in the literature that use DBpedia to generate quiz questions, such as [4, 9, 13, 14, 18]. Most of them are early demonstrators that are no longer available. Perhaps the main problem of these initiatives is the use of a simplistic question generation process, e.g. [18] and [9] only support one query type. In addition, none of them exploits Wikipedia categories and supporting images are rarely employed. A notable exception is [4] that invests more effort in the creation of question types by defining subsets of DBpedia and then generating questions (even with images). This approach for question generation is closer to the one devised in Clover Quiz, but it does not scale so well: each question type requires the extraction of a DBpedia subset, as well as changes in the quiz generation engine. In contrast, the approach of Clover Quiz is completely declarative and can be easily ported to other languages. As a result, there are more than 200K questions (in English and Spanish) built from 2.9K templates, while the other initiatives report question sets in the range of thousands.

When designing multiple choice questions, distractors have to be carefully chosen to control difficulty [7, ch. 41]. However, existing DBpedia-based question generators do not address this issue, and just use random distractors, e.g. [18], [4], and [9]. Interestingly, [2] investigates semantics-based distractor generation mechanisms, proposing several measures based on Jaccard's coefficient [10] to control the difficulty of questions and running a user study to evaluate its effectiveness. Unfortunately, [2] is limited to ontology-based questions that exploit class subsumption, so it cannot be directly applied to generate questions about entities in DBpedia. Nevertheless, Clover Quiz takes inspiration from this work to generate different lists of dis-

[17]https://en.wikipedia.org/wiki/Reliability_of_Wikipedia#Susceptibility_to_bias

[18]https://en.wikipedia.org/wiki/Ecce_Homo_(Martínez_and_Giménez,_Borja)

tractors for distinct difficulty levels, as discussed in Section 4.

On the effort required to produce the question set in Clover Quiz, the most time-consuming tasks correspond to the authoring of the domain specification files and the question templates. The former requires a close inspection of DBpedia to deal with its messiness, as discussed along Section 7. With respect to the templates, the generation of varied and high-quality questions relies on a thorough template authoring for the selected domains of interest. The approach is indeed scalable, since Clover Quiz is an individual pet project that has been fully carried out during 10 months on a part-time basis. Furthermore, an entirely automated question generation pipeline without any configuration step seems unrealistic.

Moving to system design, performance problems are experienced when submitting live queries to DBpedia, as in the case of [13]. To improve latency, some initiatives take small snapshots of DBpedia and run their own triple stores, e.g. [4] and [14], while [18] creates the question set off-line. Clover Quiz also adopts the latter approach, but it goes further by transforming the crawled data into JSON to facilitate the generation of questions, in particular to exploit Wikipedia categories. The back-end server in Clover Quiz is able to cope with the requirements of the mobile app, obtaining an average response time of $0.1s$ in a benchmarking experiment presented in Section 6.

With the release of Clover Quiz as an Android app in Google Play, more than 5K users have downloaded the game and answered more than 614K questions. User ratings are high (4.3 out of 5.0) and comments encouraging, thus suggesting that the questions generated from DBpedia are entertaining and that the game mechanics work. Future work includes the development of an iOS version. Furthermore, Clover Quiz could be extended to improve DBpedia's content through the game. Beyond Clover Quiz and DBpedia, the proposed data extraction pipeline and question generator can be used with any other semantic dataset – the only requirement is a SPARQL endpoint. As a result, a promising future line is the generation of multiple choice questions from other knowledge bases; this is especially relevant in the e-learning domain, given the importance of multiple choice questions and the advent of Massive Online Open Courses (MOOCs) [6].

Acknowledgements

This work has been partially funded by the Norwegian Research Council through the SIRIUS innovation center (NFR 237898).

References

[1] T. Alsubait, B. Parsia and U. Sattler, Generating Multiple Choice Questions From Ontologies: Lessons Learnt., in: *Proceedings of the 11th OWL: Experiences and Directions Workshop (OWLED)*, Riva del Garda, Italy, 2014, pp. 73–84.

[2] T. Alsubait, B. Parsia and U. Sattler, Ontology-based multiple choice question generation, *Künstliche Intelligenz* **30**(2) (2016), 183–188.

[3] P. Boldi and C. Monti, Cleansing wikipedia categories using centrality, in: *Proceedings of the 25th International Conference Companion on World Wide Web (WWW '16 Companion)*, Montreal, Canada, 2016, pp. 969–974.

[4] C. Bratsas, D.E. Chrysou, E. Eftychiadou, D. Kontokostas, P. Bamidis and I. Antoniou, Semantic Web game based learning: An i18n approach with Greek DBpedia, in: *Proceedings of the 2nd International Workshop on Learning and Education with the Web of Data (LiLe 2012)*, Lyon, France, 2012.

[5] T. Bray, The JavaScript Object Notation (JSON) Data Interchange Format, Proposed Standard, RFC 7159, The Internet Engineering Task Force (IETF), 2014.

[6] E. Costello, M. Brown and J. Holland, What Questions are MOOCs asking? – An Evidence-Based Investigation, in: *Proceedings of the Fourth European MOOCs Stakeholders Summit (EMOOCS 2016)*, Graz, Austria, 2016, pp. 211–221.

[7] B.G. Davis, *Tools for teaching*, 2nd edn, John Wiley & Sons, 2009.

[8] T. Heath and C. Bizer, *Linked Data: Evolving the Web into a Global Data Space*, Morgan & Claypool, 2011.

[9] B. Iancu, A Trivia like Mobile Game with Autonomous Content That Uses Wikipedia Based Ontologies, *Informatica Economica* **19**(1) (2015), 25.

[10] P. Jaccard, Étude comparative de la distribution florale dans une portion des Alpes et des Jura, *Bulletin de la Société Vaudoise des Sciences Naturelles* **37** (1901), 547–579.

[11] H. Ji, R. Grishman, H.T. Dang, K. Griffitt and J. Ellis, Overview of the TAC 2010 knowledge base population track, in: *Proceedings of the 3rd Text Analysis Conference (TAC 2010)*, Gaithersburg, MA, USA, 2010.

[12] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer and C. Bizer, DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia, *Semantic Web Journal* **6**(2) (2015), 167–195.

[13] F. Mütsch, Auto-generated trivia questions based on DBpedia data, 2017, URL: https://github.com/n1try/linkeddata-trivia, last accessed August 2017.

[14] J. Mynarz and V. Zeman, DB-quiz: a DBpedia-backed knowledge game, in: *Proceedings of the 12th International Conference on Semantic Systems (SEMANTICS 2016)*, Leipzig, Germany, 2016, pp. 121–124.

[15] L. Page, S. Brin, R. Motwani and T. Winograd, The PageRank Citation Ranking: Bringing Order to the Web, Technical Report, 1999-66, Stanford InfoLab, 1999, Previous number = SIDL-WP-1999-0120.

[16] S. Praetor, New DBpedia Release – 2016-10, 2017, URL: http://blog.dbpedia.org/2017/07/04/new-dbpedia-release-2016-10/, last accessed August 2017.

[17] W. Reese, Nginx: the high-performance web server and reverse proxy, *Linux Journal* **2008**(173) (2008).

[18] J. Waitelonis, N. Ludwig, M. Knuth and H. Sack, Whoknows? Evaluating linked data heuristics with a quiz that cleans up DBpedia, *Interactive Technology and Smart Education* **8**(4) (2011), 236–248.