User-driven Ontology Population from Linked Data Sources

Panagiotis Mitzias[1], Marina Riga[1], Efstratios Kontopoulos[1], Thanos G. Stavropoulos[1], Stelios Andreadis[1], Georgios Meditskos[1], Ioannis Kompatsiaris[1]

[1] Information Technologies Institute, Thessaloniki, Greece
{pmitzias, mriga, skontopo, athstavr, andreadisst, gmeditsk, ikom}@iti.gr

Abstract. In order for ontology-based applications to be deployed in real-life scenarios, significant volumes of data are required to populate the underlying models. Populating ontologies manually is a time-consuming and error-prone task and, thus, research has shifted its attention to automatic ontology population methodologies. However, the majority of the proposed approaches and tools focus on analysing natural language text and often neglect other more appropriate sources of information, such as the already structured and semantically rich sets of Linked Data. The paper presents PROPheT, a novel ontology population tool for retrieving instances from Linked Data sources and subsequently inserting them into an OWL ontology. The tool, to the best of our knowledge, offers entirely novel ontology population functionality to a great extent and has already been positively received according to user evaluation.

Keywords: Ontologies, OWL, Ontology Population, Linked Data, DBpedia.

1 Introduction

The rapidly increasing interest in building ontologies derives from their capability of representing knowledge in a structured and uniform way [1]. However, in order for semantically rich ontology-based applications to be deployed at an enterprise level, significant volumes of data are also required to populate the underlying models.

Populating ontologies with knowledge manually is a time-consuming and error-prone task. As a result, research has shifted attention to automating this process, introducing *ontology population*, which refers to a set of methodologies for automatically identifying and adding new instances of concepts from an external source into an ontology [2]. Ontology population does not affect the concept hierarchies and non-taxonomic relations in the ontology, leaving the structure of the ontology itself unmodified. What is affected in essence are the realisations of concepts (i.e. individuals) and the relations in the domain.

The majority of proposed tools for ontology population are aimed at NLP applications, which typically extract knowledge from natural language text and offer significant advantages over traditional export formats [3]. However, other sources of infor-

mation are very often neglected, like e.g. *Linked Data* [4], which are already more structured and semantically rich in comparison to free text.

Towards this direction, this paper presents *PROPheT* (*PERICLES*[1] *Ontology Population Tool*), a novel instance extraction engine for locating realisations of concepts (i.e. instances) in a Linked Data source, filtering them and subsequently inserting them into an OWL ontology. To the best of our knowledge and as described in the next section, no other tool currently exists that can offer the extent of functionality delivered by PROPheT.

The rest of the paper is as follows: Section 2 provides an overview of related work paradigms. Section 3 presents the proposed ontology population tool, focusing on its architecture, core components and ontology population capabilities, followed by a case study that better illustrates PROPheT's functionality. Section 5 features a user evaluation of the tool, followed by a brief discussion its limitations and directions for future work.

2 Related Work

Below is a brief account of ontology learning and population tools found in literature, mostly varying in the source of knowledge, functionality and degree of automation. For example, *DB2OWL* [5] automatically generates ontologies from relational database schemas. A mapping process detects particular cases for conceptual elements in the database and accordingly converts database components to the corresponding ontology components. The approach presented in [6] is aimed at a more efficient Linked Data consumption by proposing a learning process that can automatically construct a simple mid-level ontology, linking related ontology predicates in different data sets. Data collection is performed using SPARQL querying to the *Linking Open Data* (*LOD*) cloud. In [7], a workflow is proposed including the development of a domain ontology, the mapping between predicates in the latter and in DBpedia or other LOD sources and, finally, ontology population using SPARQL queries against DBpedia and other LOD sets. Other proposed tools conduct ontology population with knowledge derived from text documents [8, 9], spreadsheets [10] and XML files [11].

Most of the tools suggested in literature build an ontology from scratch, while our approach is aimed at enriching an existing ontology with relevant instances and property values, without any restriction to thematic domains. To the best of our knowledge, no other ontology population tool can instantiate new concepts from a LOD source so flexibly, regardless the domain of interest or the content of the source. PROPheT's flexibility lies in the fact that any kind of LOD with a served endpoint can be handled by the tool as an external source of knowledge for extracting concepts of interest and populating them to corresponding resources into the domain ontology.

Concerning the degree of automation, there is no implemented tool that carries out the whole population process automatically. PROPheT may be considered as a semi-automatic, user-driven system with different degrees of automation in various tasks:

[1] The tool has been developed in the context of the PERICLES FP7 project: http://www.pericles-project.eu/

the interaction of the software with the LOD endpoint for performing the search process is done automatically, while the final selection of the instances to be populated in the ontology and the mapping of properties needs to be done manually. The user-driven mapping process enables the dynamic and proper definition of matching elements between source and target ontologies.

3 PROPheT Overview

PROPheT is a sophisticated GUI-equipped instance extraction engine for searching instantiations of concepts in a SPARQL-served LOD source, filtering them and subsequently populating them into a local model. It offers three types of instance extraction-related functionalities, along with user-driven mapping of datatype properties. It is flexible enough to work with any OWL domain ontology and any RDF LOD set that is available via a SPARQL endpoint. PROPheT, along with documentation, is freely available at: http://mklab.iti.gr/project/prophet-ontology-populator. A detailed description of PROPheT's architecture and functionality is given subsequently.

3.1 Architecture

PROPheT's overall architecture is illustrated in Fig. 1. The front-end was implemented in Python along with PyQt [2], while specialised Python APIs (RDFLib [3], SPARQLWrapper[4]) in the back-end are deployed for handling local and remote models (ontologies) and their content. Additionally, an SQLite database was set up in the back-end for storing dynamic data (e.g. settings, user preferences) that are created during the tool's operation.

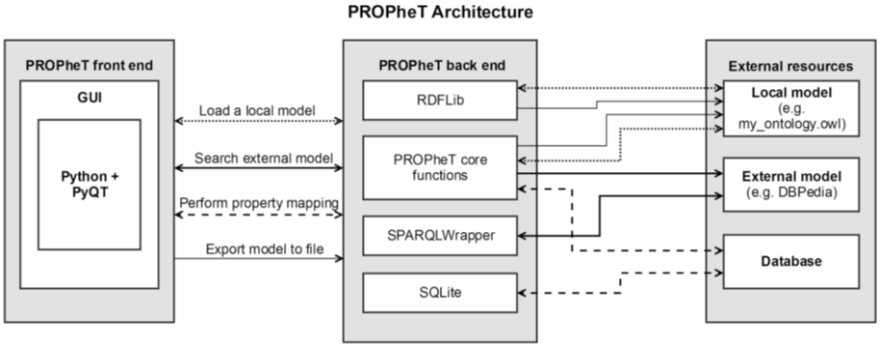

Fig. 1. PROPheT overall architecture.

[2] https://riverbankcomputing.com/software/pyqt
[3] https://github.com/RDFLib/rdflib
[4] https://github.com/RDFLib/sparqlwrapper

The arrows in the figure display key PROPheT processes; varying arrow types indicate the corresponding back-end components and external resources employed for each process.

3.2 Core Components

Below is a list of PROPheT's core components (see also workflow in Fig. 2):

- *My Model* (*MM*) is the ontology model to be populated with new instances. The ontology must comply with a specific format (.owl, .rdf, .ttl) and can reside at a local or remote location.
- *External Model* (*EM*) is the source from which new instances will be retrieved. This source should be facilitated by a SPARQL endpoint[5], so that PROPheT will be able to query directly the knowledge base via SPARQL.
- *Extraction module* (search mechanism): The current version of PROPheT features class-based and instance-based population; for more details see next subsection.
- *Mapping module*: Allows the user to match MM and EM datatype properties. When populating new instances, the module instantiates MM datatype properties with values derived from "similar" user-defined EM datatype properties.
- *Storage module* (database) stores preferences relevant to PROPheT utilities and functions, like (i) the total number of derived instances from EM, (ii) MM and EM sources, (iii) user-defined mappings, (iv) known namespaces, (v) handling of general ontology properties.
- *Export module*: A mechanism for storing the already processed/populated MM in a local file, in some of the most popular ontology file formats (*.owl, .rdf, .ttl, .nt*).

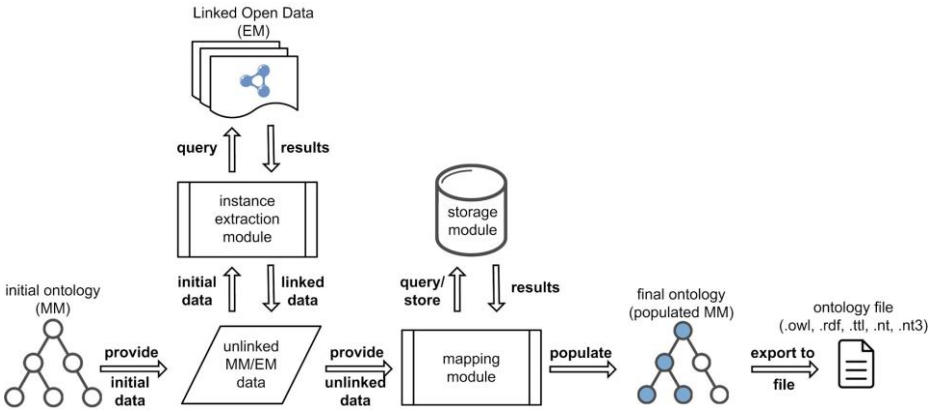

Fig. 2. PROPheT main workflow.

[5] List of SPARQL endpoints available at https://www.w3.org/wiki/SparqlEndpoints

3.3 Ontology Population

PROPheT offers the following types of instance extraction-related functionalities:

Class-based populating. This method enables the user to populate MM with new instances "from-scratch", by searching for specific types of instances in EM via entering the exact type of class, e.g. *dbo:Artist*[6]. In order to be used efficiently, the user needs to know the structure of EM or where the class type belongs in the hierarchy of the EM ontology. PROPheT submits SPARQL queries to the EM endpoint and retrieves a list of instances that belong to the specified class. The user may then select the instance(s) that he/she wishes to import (populate) under an existing MM class.

In order for PROPheT to proceed with the defined instantiation(s) for MM, a user-driven ontology mapping is performed. A list of all unique datatype properties (*owl:DatatypeProperty*) for the selected instance(s) is given to the user in order to consistently define their mapping into existing datatype properties in MM. The mapping process is further described in section 3.4.

Instance-based populating. Detecting and importing new instances via an instance-based search may be done in two different ways:

1. *Search by existing instance* - The user may select an instance already existing in MM and query the endpoint for similar instances. In particular, PROPheT performs an *rdfs:label*-based search and finds EM classes that include an instance with the desired label. The user may then select the classes of his/her interest, view their extension (i.e. set of instances) and choose which of them to import into MM.
2. *Search by instance label* - Similarly, a direct *rdfs:label*-based search is performed, with exact or partial match of the input text. However, in this case the user needs to type the desired label[7] and the search will result in a set of instances with this label, rather than similar instances of the same type (class).

In both instance-based mechanisms, the software presents a list of search results (instances) which may be selected by the user and entered into MM. As described above, a mapping process needs to take place, in order for the tool to also import the datatype properties of instances and their corresponding values.

Enrich existing instance. This type of functionality gives the user the ability to enrich an already existing instance within MM with properties and values derived from other instances in EM that have the same *rdfs:label* with the existing instance. Through this method, PROPheT performs an *rdfs:label*-based search for instances in EM that include the desired label. The derived instances may belong to one or more different classes in EM and the software tracks and presents the different *rdf:type* property declarations defined for these instances. Based on the content and semantics

---

[6] *dbo* is the prefix for a specific DBpedia URI, that is http://dbpedia.org/ontology/

[7] The search process offers options such as *Exact match*, *Contains term* and case sensitivity.

of the derived instances, the user may decide which property-value pair(s) he/she wishes to import into MM for the initially selected instance. Similarly to the aforementioned functions, an ontology mapping process should be performed in order for the new properties and values to be added to the existing instance.

3.4 Mapping Classes and Properties

While populating a model, the user is required to manually define mappings between MM and EM classes and properties. In particular, the user needs to specify the MM class where instances extracted by any of the three above processes will be imported. Additionally, a mapping between EM and MM datatype properties is considered fundamental in order to insert datatype property values to the latter. For example, the user might define that the EM property *dbo:birthDate* corresponds to the MM property *:dateOfBirth*. Once defined by the user, PROPheT memorizes such mappings and offers suggestions when the need for the same mappings reappears. The local model may also be semantically enriched by optionally affiliating EM and MM properties via relation *owl:equivalentProperty* and classes via *owl:sameAs* or *rdfs:seeAlso*. Additional associations, like e.g. *skos:narrower* and *skos:borader* from SKOS [12] are considered for the next version of the tool.

4 Case Study

This section illustrates PROPheT's functionality through specific examples that involve extracting information from two different LOD sources: DBpedia and LinkedMDB[8]. Consider a case where the user wants to populate an existing ontology containing information regarding artists (*ex:Artist*) and artworks (*ex:Artwork*). More specifically, let's suppose the user wishes to add the movie "*The Godfather*" into *ex:Artwork*. In order to use PROPheT, the user loads his/her model and registers LinkedMDB as the current EM source.

Since the name of the movie is specified a priori, he/she could search for instances through the "*Search by Instance Label*" method: when typing the movie title only one result is retrieved that corresponds to the entry of this movie in LinkedMDB[9]. Information retrieved for that entry is limited, due to the fact that PROPheT handles only datatype properties and their values, and not the attached object properties; on the other hand, LinkedMDB contains mostly object properties that connect instances of different types of classes.

Continuing the case study, suppose that the user wishes now to take advantage of information stored in other LOD sources, like for example in DBpedia. In this case, he/she has to modify the currently selected EM in PROPheT to DBpedia, and then to apply the "*Enrich existing instance*" method. PROPheT submits a SPARQL query to DBpedia and retrieves a set of instances that may belong to different classes, but they

[8] http://www.linkedmdb.org/

[9] The exact entry in LinkedMDB is http://data.linkedmdb.org/resource/film/43338

all share the same *rdfs:label* with the newly populated instance in MM. At that point, the user may select any pair(s) of datatype properties/values he/she wants to add to the MM instance of "*The Godfather*" movie. After manually mapping the relevant EM and MM properties, the data is inserted into the corresponding fields in the MM ontology. An indicative screenshot of the populated ontology in PROPheT thus far, can be seen in Fig. 3.

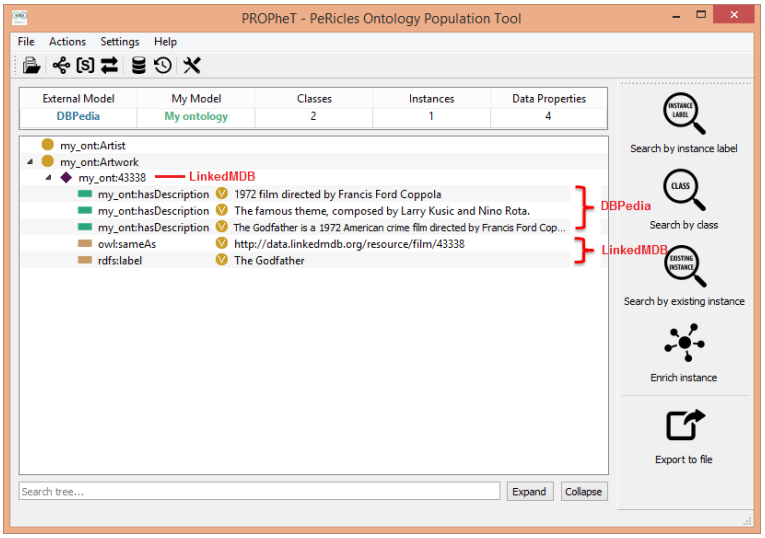

Fig. 3. PROPheT's main window with newly populated instance in class *ex:Artwork*; indicative values in properties are presented, retrieved from DBpedia and LinkedMDB.

In case the user wishes to search for and retrieve similar resources that belong to a specific class, then he/she can employ PROPheT's methods "*Search by Class*" or "*Search by Existing Instance*" for any selected endpoint. For instance, if the former method is selected, the user has to type the exact *prefix:class_name* that he/she is interested in, e.g. *dbo:Film* for DBpedia, or *movie:film* for LinkedMDB. A set of instances will be retrieved, and the user may then proceed with the selection and mapping process as described previously.

If, on the other hand, "*Search by Existing Instance*" is selected, PROPheT will take into account the predicate-object pair (<*rdfs:label* '*The Godfather*'>) of the newly populated instance (*ex:43338*[10]) to search for alternative classes that contain instances with the same label. When applying this search in DBpedia, numerous classes are retrieved, specified either as internal classes via DBpedia URIs or as external classes via URIs originating from adopted ontologies (e.g. *YAGO*[11], *SKOS*[12], *Wikidata*[13], etc.).

[10] Name of instance was retrieved from instance of "*The Godfather*" movie in LinkedMDB.
[11] www.mpi-inf.mpg.de/YAGO/
[12] https://www.w3.org/2008/05/skos
[13] https://meta.wikimedia.org/wiki/Wikidata/Development/RDF

An interesting aspect is the diversity of types of the retrieved instances, that range from more generic (e.g. *skos:Concept*, *dbo:Work*) to more specialized concepts (e.g. *yago:1970sCrimeFilms*, *yago:1972Films*, *yago:FilmsBasedOnNovels*, *wikidata:Q11424*, etc.). The user may select one or more classes from which instances will be retrieved and proceed with the selection of instances to be populated (see Fig. 4) and with the mapping process. Consequently, with this method we achieve integrating instances from *different* DBpedia classes into *one single* MM class.

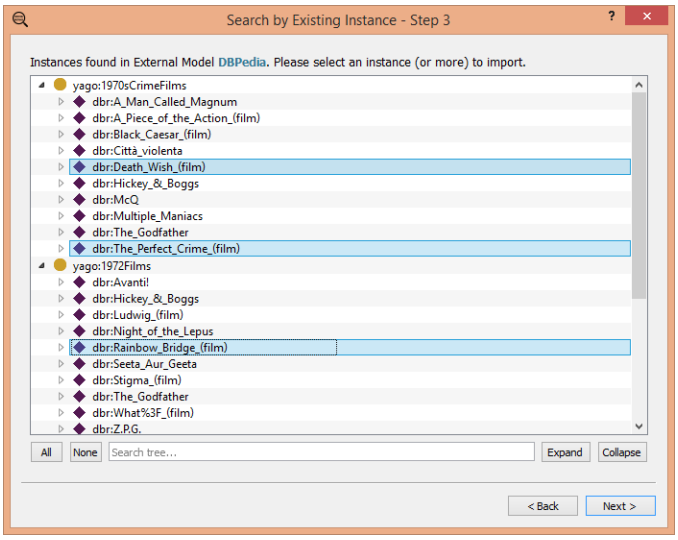

Fig. 4. Selection of instances from different classes (declared via *rdf:type*) where the newly populated instance belongs to.

5 User Evaluation

The tool's user evaluation involved 15 participants, aged above twenty, with a computer science background (either BSc, MSc or PhD), most of them (~80%) familiar and experienced with Semantic Web technologies (RDF, OWL). The trial was performed remotely at the convenience of each participant, as they were provided with a link to download the tool and access the online user guide. Initially, users had to fill in demographic details and perform a set of predefined tasks in the form, and answer based on the outcomes.

As the tasks required full grasp of the tool's function, even when consulting the user guide, the answers were used simply to validate a user's eligibility, which turned out to be 100%. Rating the system with respect to different aspects, in a scale from 1 to 7, where 1-3, 4, 5-7 are considered a positive, neutral and negative answer respectively, yielded positive results in all domains, shown on Fig. 5.

Due to space restrictions and the currently small user sample size, a more extended evaluation, using a universal questionnaire, (e.g. SUS [13]), is planned as future work.

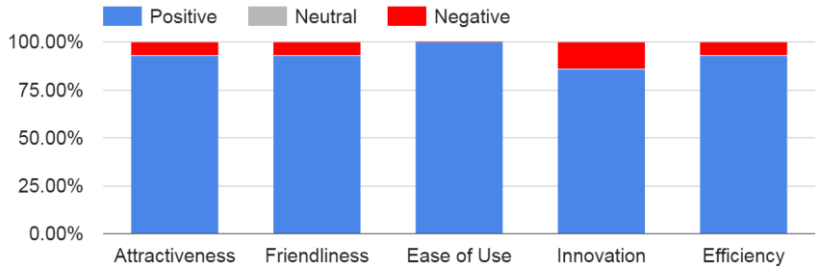

Fig. 5. User evaluation results.

6 Limitations and Future Work

The type of populated information is significant for the completeness of an ontology population tool. Instances in ontologies may contain values for both *object properties* (*owl:ObjectProperty*) that interrelate individuals, and *datatype properties* (*owl:DatatypeProperty*) that relate individuals to literal values. In PROPheT's current version, population is limited to datatype properties only; relationships between instances are significantly more complex since each property is limited by domain and range declarations and the manual alignment between MM and EM properties would most probably lead to inconsistent results.

Furthermore, the current version of the tool does not handle direct or indirect imports of modular ontologies. Our aim in the future versions is to handle the MM ontology as an extended version of an ontology-graph, where declarations of triples of imported ontologies will be combined with those triples that are defined only in the domain ontology.

A further limitation is that PROPheT does not exploit the structure of the ontology that the user wants to populate: an alignment of the internal ontology with the external ones could save lots of effort and increase the usefulness of the system.

Last, the problem of instance redundancy (i.e. two or more instances in the ontology refer to the same real object) is handled by PROPheT in a way that instances with the same name-identifier cannot be populated multiple times in the ontology, i.e. values of populated data properties are linked to one instance. As future work, PROPheT may encompass more complex handling mechanisms, such as heuristics or machine learning methods to identify similar resources.

7 Conclusions

The process of ontology population is a non-trivial but also laborious task, while the relevant proposed approaches are typically focused on analysing natural language text, often overlooking other sources of more structured information, like e.g. Linked Data. This paper presents PROPheT, a novel user-driven ontology population tool for semi-automatically retrieving instances from a Linked Data source and inserting them into an OWL ontology. Through the use of embedded wizards, the user may apply

advanced class-based and instance-based queries to the LOD endpoint, without any precondition in knowing technical details of the applied queries or of the SPARQL query language's syntax. The embedded mapping process enables the dynamic and proper definition of matching elements (i.e. classes and properties) between source and target (populated) ontologies. The extent of functionality and flexibility offered by the tool cannot be matched by any other software currently found in literature, making PROPheT a truly novel system for populating ontologies.

Acknowledgments. This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 601138.

References

1. Stephan, G. S., Pascal, H. S., & Andreas, A. S. (2007). Knowledge representation and ontologies. *Semantic Web Services: Concepts, Technologies, and Applications*, 51-105.
2. Buitelaar, P., & Cimiano, P. (2008). *Ontology learning and population: bridging the gap between text and knowledge* (Vol. 167). Ios Press.
3. Petasis, G., Karkaletsis, V., Paliouras, G., Krithara, A., & Zavitsanos, E. (2011, January). Ontology population and enrichment: State of the art. In *Knowledge-driven multimedia information extraction and ontology evolution* (pp. 134-166). Springer-Verlag.
4. Bizer, C., Heath, T., Idehen, K., & Berners-Lee, T. (2008, April). Linked data on the web (LDOW2008). In *Proc. 17th Int. Conf. on World Wide Web* (pp. 1265-1266). ACM.
5. Ghawi, R. and Cullot, N. (2007). Database-to-Ontology Mapping Generation for Semantic Interoperability. In *VLDB '07 Conf.*, VLDB Endowment ACM, pp. 1-8, Vienna, Austria.
6. Zhao, L. and Ichise, R. (2012). Mid-Ontology Learning from Linked Data. In *The Semantic Web*, pp. 112-127, Springer Berlin Heidelberg.
7. Gavankar, C., Kulkarni, A., Fang Li, Y. and Ramakrishnan, G. (2012). Enriching an Academic Knowledge base using Linked Open Data. In *Proc. Workshop on Speech and Language Processing Tools in Education in 24th Int. Conf. on Comput. Linguistics*, pp. 51-60.
8. Maynard, D., Funk, A. and Peters, W. (2009). SPRAT: A tool for automatic semantic pattern-based ontology population. *In Int. Conf. for Digital Libraries and the Semantic Web*, Trento, Italy.
9. Velardi, P., Navigli, R. and Missikoff, M. (2002). Integrated approach for Web ontology learning and engineering. *In IEEE Computer*, Vol. 35, No. 11, pp. 60-63.
10. Han, L., Finin, T., Parr, C., Sachs, J. and Joshi, A. (2008). RDF123: from Spreadsheets to RDF. *In 7th Int. Semantic Web Conference*, pp. 451-466, Springer Berlin Heidelberg.
11. Modica, G., Gal, A. and Jamil, H.M. (2001). The Use of Machine-Generated Ontologies in Dynamic Information Seeking. *In Cooperative Information Systems*, pp. 433-447, Springer Berlin Heidelberg.
12. Miles, A., & Bechhofer, S. (2009). SKOS simple knowledge organization system reference. *W3C recommendation*, *18*, W3C.
13. Brooke, J. (1996). SUS-A quick and dirty usability scale. Usability evaluation in industry, 189(194), 4-7.