Proceedings of the 4th World Congress on Electrical Engineering and Computer Systems and Sciences (EECSS'18)
Madrid, Spain – August 21 – 23, 2018
Paper No. CIST 101
DOI: 10.11159/cist18.101

A Specialized Recommender Agent for the Semantic Web

Neli P. Zlatareva
Central Connecticut State University
1615 Stanley Street, New Britain, CT 06050, USA
zlatareva@ccsu.edu

Abstract **-** Information overload is one of the main challenges for web users nowadays. Intelligent Personal Assistants provide some help in dealing with this issue by navigating and collecting information, but they are not capable of meaningfully integrating and interpreting that information to address personalized user queries. Transition to the Semantic Web promises a dramatic shift in the scope of web services and the way they are performed. The so-called *Semantic Recommender Systems* utilize Semantic Web technologies to carry out a highly focused search based on similarities between concepts. This paper discusses a specialized recommender agent which derives contextual knowledge from targeted search on a simulated linked data network to provide personalized recommendations in the domain of interest. An extended example is followed throughout the paper to illustrate the type of queries the agent is intended to address.

Keywords**:** Semantic Web, Information Retrieval, Linked Data, Web Search, Emerging Applications, Knowledge Representation.

1. Introduction

The latest advances in Semantic Web technologies are getting us closer to materializing Tim Berners-Lee's vision of the web as an "… environment where software agents roaming from page to page can readily carry out sophisticated tasks for users" [1]. The so-called *Intelligent Personal Assistants* (IPA) are changing the web culture by providing dramatically extended and improved web services. They can automatically perform management tasks by initiating search from various web resources and databases in response to user queries. *Siri*, the famous Apple's personal assistant, does an impressive job answering queries about movies, restaurants, weather reports, etc. Google's *Google Now* has similar capabilities and its latest version, *Google Assistant*, has sophisticated natural language capabilities allowing it to engage in a two-way dialog with the user. Although a huge commercial success, these systems are intended to help the general user navigate and collect information, but they are not capable of meaningfully integrating and interpreting that information to address personalized queries. Consider, for example, the following scenario. A transfer student is looking for a school which would accept most of her credits, offers a program of interest which allows her to graduate on time, and satisfies most of the student's preferences related to the location of the school, financial aid, etc. IPAs are not suited to deal with this type of complex descriptive queries and the user is left to search the "old-fashioned way", i.e. she will browse a (limited) number of universities' web sites to gather relevant information which she will evaluate in possibly ad-hoc manner to make a decision. Given the variety of choices and considerably different structure and terminology used by those sites, this may become a tedious and unproductive process. Assume she is starting her search by looking for a computer science program offered by a public university in Connecticut. Google search will return millions of hits, none of which will be specific enough. If the search were conducted on the *linked data* network, the result would have been much more focused and helpful. The sharp contrast between the current "linked documents" web (commonly referred to as *web 2.0*) and the emerging linked data web, or *web 3.0*, was illustrated by Tim Berners-Lee in his TDC 2009 conference speech [2], where he gave the following example: an attempt to google proteins involved in signal transduction that are also related to pyramidal neurons, resulted in 223,000 unhelpful hits, while the same query posted on a linked data network returned 32 highly relevant hits.

It is widely acknowledged that future advances in web services rely on open access data stores and novel AI-based search techniques able to efficiently identify, integrate, and process relevant data to provide personalized recommendations. *Semantic Recommender Systems* (SRSs) go one step further the commercial IPAs in addressing this need. A SRS according to [3] is " … any system that bases its performance on a knowledge base, normally defined through conceptual maps (like a taxonomy or thesaurus) or an ontology, and that use technologies from the Semantic Web." Traditional recommender

systems are intended to provide users with recommendations of products and services by keeping track of their past activities utilizing users' "interest scores" and items' ratings, and typically employ common collaborative filtering techniques [4]. These techniques, however, are not efficient if there is not enough information about the user or the product, in which case the so-called *closed recommendation systems* [5] that rely on additional user data collected and stored in a private data base can be used to enhance the recommendation process. An alternative approach is suggested in [6], where user data is obtained from open data sources on the Linked Open Data (LOD) project. Although such an advancement makes it possible to easily share user data among multiple applications, it is of little help if the user profile is not expected to affect the recommended outcome if it depends on a highly specialized contextual situation like the example scenario described above. In this paper, we argue that generating recommendations to address personalized user queries requiring specialized domain knowledge can be carried out by a semantic recommender agent utilizing contextual knowledge from a targeted search on a linked data network. We discuss how linked data can be used by the agent to tackle application tasks as the one described above.

The paper is organized as follows. First, we discuss the preliminaries and the motivation for our research, followed by a description of a query guided search carried out on a linked data network. How collected information is used to build the context where user queries are interpreted and executed is discussed in Section 4. The example scenario is followed throughout the paper to illustrate the type of queries that our recommender agent is intended to handle.

2. Preliminaries and Motivation

Consider again our example scenario. Assume that universities, in addition to their traditional web sites, also support *Resource Description Framework* (RDF) (https://www.w3.org/RDF/) data stores, known as *triple stores*, containing the same information for use by web agents. Then, the search for a school matching user description can be delegated to a software agent which will autonomously navigate universities' triple stores collecting and processing relevant information to deliver a recommendation for the user.

There are two interrelated tasks that the agent must be able to carry out to tackle this job:

1.It must be able to augment its background knowledge with query specific information derived from visited triple stores to build the context in which user query will be interpreted. In [7], we have introduced a reasoning framework utilizing context-dependent rules which aimed to accommodate imperfect data to reason about a specific goal as part of the user's query. It was assumed that the rules were already acquired from the data collected from the linked data network, which is a separate task as defined next.

2.The agent must be able to initiate and carry out search on the linked data network to collect relevant information for building the context where user query will be interpreted, and convert this information into a logical representation to "run" the query.

This paper mainly addresses the second task. The technologies at hand the recommender agent can utilize to carry out this task are the *Web Ontology Language* (OWL) (http://www.w3.org/TR/owl-features/) for knowledge representation and the *SPARQL Protocol and RDF Query Language* (SPARQL) (http://www.w3.org/TR/rdf-sparql-query/) for information retrieval. The latest version of OWL, OWL 2 (http://www.w3.org/TR/owl2-profiles/), was developed to support scalability and efficiency of Semantic Web search by further restricting the expressivity of the language. One of the profiles of OWL 2, OWL 2 RL, was especially designed to support rule-based reasoning in large datasets stored as RDF triples. The set of rules governing this process is based on RDF direct semantics, which suggests that OWL inference capabilities are strictly monotonic. SPARQL on the other hand, can refer to entities that do not exist in the triple store, because such entities can be ignored during the search process, i.e. it operates under the *Open World Assumption* (OWA) and therefore the returned results may be incomplete. This suggests that the recommender agent should be able to reason with incomplete data, or in some cases inconsistent data if the original query contains contradictory statements.

Dealing with incomplete, uncertain, and/or inconsistent knowledge in the context of the Semantic Web has been an active research area for a number of years. Traditional artificial intelligence approaches to non-monotonic reasoning are not applicable in the context of the Semantic Web because of their reliance on the *Closed World Assumption* (CWA). Latest efforts to address this issue are mostly concerned with the development of probabilistic extensions of OWL [8, 9]. However, extending the representational power of OWL in such a way does not guarantee that reasoning with probabilistic OWL ontologies will be efficient or even decidable. A more feasible approach is taken in [10] where the underlying representation, the *knowledge graph*, provides contextual meaning to the search and is enhanced by the provenance of each fragment of knowledge captured used to compute their confidence probabilities. The logical framework based on context-dependent rules

discussed in [11] has a similar motivation. It was shown there that adding contextual information to deal with incompleteness and/or uncertainty in a limited context defined with respect to the user's query, offers a practical approach for implementing a recommender agent capable to not only derive a recommendation but also to justify it by providing the context where this recommendation holds. Next, we show how this framework can be extended to acquire query specific context-dependent rules from the context derived from a linked data network. For that, we have built two triple stores mirroring selected parts of two university web sites [12]. Each triple store was created according to its own ontology to simulate the real-world case where terminological consistency between independent datasets cannot be ensured. Triple stores were implemented in Turtle (https://www.w3.org/TR/turtle) and ontologies were built in Protégé (http://www.protege.stanford.edu). We used Jena SPARQL query engine ARQ (http://jena.apache.org/documentation/query/) to collect and integrate information from the triple stores.

3. Query Guided Search on a Linked Data Network

Assume as part of our example scenario, that the student is looking for a program where she can transfer her "Computer Programming", "Data Structures" and "Data Bases" courses. The student also wants to graduate in two years and wants to know where in the program these transferred classes will place her. It is important to note that since there is no standard vocabulary in the "university domain", schools can use different terminology to refer to the same web resource, and vice versa, the same name can be used for different resources. That is, Semantic Web operates under the *Non-Unique Name Assumption* and instead of being defined by name, resources are defined in terms of their properties. To find whether a course named "Data Structures" is the same as a course with unknown name in a different school, the two course descriptions are supposed to match. But even if the courses are indeed "the same" their descriptions may not match exactly (they may cover different "elective" topics or use different tools to teach the same topics, for example).

Retrieving information from a linked data network can be carried out by an appropriate SPARQL query. SPARQL is a very rich language offering four different query forms depending on the expected result. The SELECT query uses pattern matching algorithm to retrieve specific information, while DESCRIBE and CONSTRUCT queries return RDF graphs that can be combined with the agent's background knowledge for further processing. The DESCRIBE query is especially useful in our case, because the agent is not expected to know what it is searching for, but it should know what it wants to learn about (in our case "tell me everything about all the courses that cover programming, data structures, and data bases"). The DESCRIBE query can be used to search for information about these implicitly defined resources guided by its own processor, rather than by the query itself.

To collect relevant information from the triple stores, the recommender agent must initiate a number of SPARQL queries. For that, it must have some background knowledge about what type of query will be useful at each step of the search process. This knowledge can be embedded in a library of empty query frames and a set of domain specific rules on how to populate these frames according to user description. For example, searching for a school offering a computer science program will need a SELECT query, while searching for course descriptions to compare courses will need a DESCRIBE query. Following up with our example scenario, assume that the student is looking for a computer science program that would allow her to graduate in 2 years given that the three courses she already took can be transferred into the program. The SELECT query will return a list of programs of interest together with all of the relevant information that is available for them (note below that the returned information is incomplete for some of the programs). Because SPARQL is not only a query language, but also a protocol, the results of the query can be returned in XML format, which can be further converted into a selected RDF serialization.

Assume that the following result is returned by the SELECT query (shown here in text format for readability):

univ	program	prog_type	req_cred	elect_cred	total_cred
CCSU	CS	Minor			18
CCSU	CS Alt	BS			53
CCSU	CS Hon	BS			79
ECSU	CS MIS	Minor	9	9	18
ECSU	CS CEng	Minor	17	3	20
ECSU	CS Bioinf	Minor	19	3	21

```
|ECSU |CS        | Minor    | 9         | 6         | 15        |
|ECSU |CS        | BS       | 37        | 12        | 49        |
-------------------------------------------------------------------------
```

There are three programs matching student description. To further evaluate these programs, the agent needs to decide whether the courses taken by the student can be transferred into any of these programs. Notice that two of the programs are offered by the same school and therefore the same DESCRIBE query can be used to retrieve information to evaluate the potential match. The DESCRIBE query below returns a Turtle file containing all the information that might be relevant to the course transfer request.

```
DESCRIBE ?x
where {?x rdf:type univ:Course .
            ?x univ:aboutSubject ?subject.
                filter((regex(str(?subject),"Programming")) ||  (regex(str(?subject), "DataStructures" )) ||
                      (regex(str(?subject), "DataBase" )))}
```

Here is an excerpt of the Turtle file returned by the query to the “CCSU” triple store:

```
data:CS253  a    univ:Course ;
              univ:aboutSubject     data:DataStructures ;  univ:courseName     "Data and File Structures" ;
              univ:coversCoreTopic     data:AlgorithmAnalysis ,  data:Graphs , data:Trees , data:Sorting ;
              univ:coversElectiveTopic    data:RecurenceRelations ;  univ:hasPrerequisiteCourse  data:CS153 ;
              univ:hasRecommendedPrerequisiteCourse data:MATH218 ;
              univ:hasRequiredPrerequisiteCourse data:CS152 ;  univ:numberCredits 3 .
data:CS152  a    univ:Course ;
              univ:aboutSubject    data:ObjectOrientedProgramming , data:DataStructures ;
              univ:courseName   "Computer Science II" ;
              univ:coversCoreTopic  data:LinkedLists, data:Inheritance, data:EventDrivenProgramming,
                                          data:Recursion,  data:Searching ;
              univ:coversElectiveTopic  data:Queues , data:Sorting ,  data:Stacks ;
              univ:coversTopic   data:ObjectOrientedProgramming ;
              univ:hasRecommendedPrerequisiteCourse  data:MATH221 ;
              univ:hasRequiredPrerequisiteCourse  data:CS151 ;   univ:numberCredits  3 .
data:CS464  a    univ:Course ;  univ:aboutSubject     data:ProgrammingLanguages ;
              univ:courseName       "Programming Languages" ;
              univ:coversTopic      data:ProgrammingLanguageComparison ;
              univ:hasPrerequisiteCourse  data:CS253 ;   univ:numberCredits  3 .
data:CS153  a     univ:Course ;  univ:aboutSubject    data:ComputerProgramming ;
              univ:courseName     "Computer Science III" ;
              univ:coversTopic    data:DataStructures_Topic, data:IntegratedDevelopmentEnvironments,
                                          data:TeamSoftwareDevelopment;
              univ:hasPrerequisiteCourse  data:CS152 ;   univ:numberCredits 3 .
data:CS460  a     univ:Course ;   univ:aboutSubject     data:DataBases ;
             univ:courseName    "Database Concepts" ;
             univ:coversTopic   data:DatabaseDesign ,  data:DatabaseUsage ;
             univ:hasPrerequisiteCourse  data:CS253 ; univ:numberCredits  3 .
data:CS151  a     univ:Course ;
             univ:aboutSubject     data:ObjectOrientedProgramming ;
             univ:courseName    "Computer Science I" ;
            univ:coversCoreTopic  data:Classes , data:Selection , data:Iteration ,   data:DataTypes ;
            univ:coversElectiveTopic  data:Inheritance ;
            univ:coversTopic   data:Iteration , data:DataTypes ,  data:ObjectOrientedProgramming ;
            univ:hasRecommendedPrerequisiteCourse data:CS113 ;
            univ:hasRequiredPrerequisiteCourse data:MATH152 ;   univ:numberCredits 3 .
```

Notice that there is no fixed pattern that course descriptions follow which is consistent with the "*Anyone can say Anything about Any topic*" (AAA) slogan underlying the Semantic web ([13]. Some descriptions are more specific discriminating between "required" and "recommended" prerequisites, or "core" and "elective" topics. Comparing their similarity to user data (courses the student is looking to transfer), therefore, must be carried out in the context of these descriptions. To do that, these descriptions must be converted into a set of context-dependent rules which as shown next provide an efficient way to reason about similarities and differences between user data and acquired descriptions.

4. Acquisition of Context-Dependent Rules from Turtle Files

Context-dependent rules were introduced as a way to handle incomplete and/or inconsistent knowledge under the OWA [11]. They employ two types of premises which allow them to discriminate between certain and missing/uncertain data. Their general format is as follows:

R_i: $(SP_1 \ \ldots \ SP_n)$ $(WP_1 \ \ldots \ WP_m) \rightarrow A(L)$, where

- $<R_i>$ is a rule reference;
- $<(SP_1 \ \ldots \ SP_n)$ $(WP_1 \ \ldots \ WP_m)>$ is the body of the rule consisting of two types of premises depending on the strength of their support for the rule's conclusion. We refer to them as *strong premises* and *weak premises*, respectively. All rule premises must be positive statements to be matched to RDF triples.
- <A> is the conclusion of the rule, and <L> is a label matching A to the following set of truth values {T/True, F/False, S/Supported}. If $\neg$A(T) is derived (for example, in case where the user is explicitly prohibited from taking a course she is inquiring about), then an inverse statement A(F) is added to the data set. A negated conclusion can only be used to identity a contradiction if both A(F) and A(T) are present, but it cannot be used as a rule premise.

The conclusion, A, can be derived in the following two contexts:

1. All strong premises hold, but none or some of the weak premises hold. The set of strong premises alone defines the minimal context for the conclusion to be considered "possibly sound". If any of the weak premises also hold, the minimal context is extended with those weak premises thus increasing the nominal support of the conclusion. Note that the notion of "soundness" can be used to justify the agent's recommendation if needed.
2. All strong premises and all weak premises hold. This defines the maximal support for the conclusion and the label associated with it is T. This is also the case if the set of weak premises is empty. The latter defines a monotonic rule, while the other cases define non-monotonic rules. Contrary to non-monotonic rules working under the CWA, where non-monotonic premises serve as "exceptions" preventing rule firing, in context-depending reasoning weak premises reflect the degree of soundness/belief of the conclusion.

Recall that the DESCRIBE query initiated by the agent aimed to retrieve information about courses that satisfy certain property, in our case *aboutSubject*. Although the agent does not know what those courses are going to be, it must know how the result of the search will be used in addressing user query, which in our case will be whether a course is transferable and where in the program the student will be placed in. That is, the agent must be able to derive the support for the following two relations: *isTransferableAs* and *canBeTaken*. For that, it must have some background knowledge about how to use the data from the Turtle file to derive statements satisfying these relations. This background knowledge can be defined as a set of domain specific "meta-rules" used to guide the acquisition of context-dependent rules directly or indirectly connecting available data to the relations of interest.

The following "meta-rules" are used to acquire context-dependent rules from the Turtle file to support *isTranferableAs* and *canBeTaken*:

$MR_{isTranferableAs}$: [(? course univ:aboutSubject ? subject) (?course univ:numberCredits ?num)
[(?course univ:coversCoreTopic ? ctopic)$_i$]] [[(? course univ:coverElectiveTopic ? etopic)$_j$]] →
→ (dataS:Course univ:isTranferableAs ? course) (? support))

$MR_{canBeTaken}$: [(? course univ:hasRequiredPrerequisiteCourse ? prereq)$_i$]
[(? course univ:hasRecommendedPrerequisiteCourse ? prereq)$_j$] →
→ (? course univ:canBeTaken dataS:Student) (? support))

According to $MR_{canBeTaken}$, the following rule can be derived from the Turtle file:

R_i: ((data:CS253 univ:hasRequiredPrerequisiteCourse data:CS152))
((data:CS253 univ:hasRecommendedPrerequisiteCourse data:CS153)

(data:CS253 univ:hasRecommendedPrerequisiteCourse data:MATH218)) →
→ ((data:CS253 univ:canBeTaken dataS:Student) (S))

This rule states that Student can take CS 253 if she has CS152, but having Math 218 and CS 153 although not required is relevant to the conclusion, CS 253, and can provide an additional support for it. The truth value, *S*, reflects the uncertainty about how this conclusion should be interpreted by the agent in the further decision-making process.

Definition. The set of rules acquired from the Turtle file comprise the initial rule set, $R\text{-}set_0$, and defines the benchmark context relevant to the user query.

To implement the idea of context-dependent reasoning, $R\text{-}set_0$ is extended with the so-called *duplicate rules* [11] as follows: for each $R_i \in R\text{-}set_0$, a set of duplicate rules of the form R_i^*: $(SP_1 \ \dots \ SP_n \ WP_{i1} \dots WP_{ik})$ $(\{WP_1 \ \dots \ WP_m\} \setminus \{ WP_{i1} \dots WP_{ik}\}) \rightarrow A(L)$, where $\{i_1, \dots, i_k\} \subseteq \{1, \dots, m\}$ is generated.

Definition. $R\text{-}set_{ext} = R\text{-}set_0 \cup \{R_i^*\}_j$ defines the extended query specific context where the user description will be interpreted.

In our example, the following three duplicate rules will be associated with *canBeTaken*:

R_i^*: ((data:CS253 univ:hasRequiredPrerequisiteCourse data:CS152)
(data:CS253 univ:hasRecommendedPrerequisiteCourse data:CS153))
((data:CS253 univ:hasRecommendedPrerequisiteCourse data:MATH218)) →
→ ((data:253 univ:canBeTaken data:Student) (S))

Note that although the truth value of this duplicate's conclusion is the same, *S*, the context in which the conclusion is derived is extended with an additional justification, thus increasing the agent's confidence in the conclusion. Similarly,

R_i^{**}: ((data:CS253 univ:hasRequiredPrerequisiteCourse data:CS152)
(data:CS253 univ:hasRecommendedPrerequisiteCourse data:MATH218))
((data:CS253 univ:hasRecommendedPrerequisiteCourse data:CS153)) →
→ ((data:253 univ:canBeTaken data:Student) (S))

Finally, assume that the student has both CS 153 and MATH 218. The following version of this rule defines the maximal context, where the conclusion must necessarily hold:

R_i^{***}: ((data:CS253 univ:hasRequiredPrerequisiteCourse data:CS152)
(data:CS253 univ:hasRecommendedPrerequisiteCourse data:CS153)
(data:CS253 univ:hasRecommendedPrerequisiteCourse data:MATH218)) ( ) →
→ ((data:253 univ:canBeTaken data:Student) (T))

Given $R\text{-}set_{ext}$, the agent can match user description against it. Notice that the match needs not be exact; this is the advantage of duplicate rules. It should be noted that popular OWL reasoners such as Fact++ (http://owl.man.ac.us/factplusplus) and Racer (http://www.racer-systems.com) cannot adequately handle this case because their inference is strictly monotonic.

To follow up with our example, consider the "Data Structures" course that the student wants to transfer. The formal description of this course is assumed to be available in the triple store of the school that the student attended. For the sake of simplicity, let us assume that both triple stores use the same terminology (ultimately, we want to have a standard terminology across the entire "university domain") and the "Data Structures" course taken by the student is defined as follows:

dataS:CS105 a univS:Course ;
univS:aboutSubject dataS:CProgramming , dataS:DataStructures ;
univS:courseName "Data Structures" ;
univS:coversCoreTopic dataS:LinkedLists, dataS:EventDrivenProgramming,
dataS:Recursion, dataS:Searching, dataS:Sorting ;
univS:hasRequiredPrerequisiteCourse dataS:CS104 ; univS:numberCredits 3 .

To check whether CS 105 matches any course from the Turtle file, the agent must show that:

$R\text{-}set_{ext} \cup$ {dataS:CS105 univS:aboutSubject dataS:CProgramming, dataS:DataStructures ;
univS:coversCoreTopic dataS:LinkedLists, dataS:EventDrivenProgramming,
dataS:Recursion, dataS:Searching, dataS:Sorting.} →
→ (dataS:CS105 univ:isTransferableAs ? course) (? support)

The following backward-chaining procedure will do that:

1. Identify the set of "strong" premises of all *isTransferableAs* rules with the same *aboutSubject* property.
2. For each rule from that set, consider all triples with *coversCoreTopic* property (these are all strong premises) and match them to the triples from the query.
    a. If all such premises match, mark the conclusion, *isTransferableAs*, as *S* (supported).
    b. For those premises that do not match, generate a DESCRIBE query to find courses in the student's school that are prerequisites for CS105. If such courses are found and they are included in the student description, mark *isTransferableAs* as *S*.
3. For all supported conclusions, find the rule from $R\text{-}set_{ext}$ providing the maximum support for the conclusion. This rule defines the extended context for the conclusion and the value for (? support). All other rules deriving the same conclusion are ignored.

In our example, this procedure will return the conclusion (dataS:CS105 univ:isTransferableAs data:CS152) (S) with the following extended context associated with it

((data:CS152 univ:coversCoreTopic data:LinkedList) (data:CS152 univ:coversCoreTopic data:Inheritance)
(data:CS152 univ:coversCoreTopic data:EventDrivenProgramming)
(data:CS152 univ:coversCoreTopic data:Recursion) (data:CS152 univ:coversCoreTopic data:Searching)
(data:CS152 univ:coversElectiveTopic data:Sorting))

5. Conclusion

The paper outlined a general framework of a recommender agent which makes use of Semantic web technologies to provide personalized recommendations in a specialized problem domain. We discussed how the agent interacts with the linked data network to search for query-specific information, and how it converts the results of the search into context-dependent rules to "run" the query. To implement the presented framework, we have utilized two independent triple stores mirroring selected parts of two universities' web sites. Each triple store was created according to its own ontology to simulate the real-world case where terminological consistency between independent datasets cannot be ensured. We discussed how context-dependent reasoning facilitates query interpretation and execution. An extended example was used throughout the paper to illustrate the type of queries the recommender agent is intended to handle.

Acknowledgements

This work was partially supported by a CSU-AAUP research grant.

References

[1] T. Berners-Lee, (2006), *The Semantic Web Revisited*. [Online]. Available: http://eprints.soton.ac.uk/262614/1/Semantic_Web_Revisted.pdf

[2] T. Berners-Lee (2009) *The Next Web. TDC 2009 conference speech.* [Online]. Available: http://www.ted.com/talks/tim_berners_lee_on_the_next_web.html

[3] E. Peis, J. M. Morales-del-Castillo, and J. A. Delgado-López, (2012), Semantic Recommender Systems. *Analysis of the state of the topic.* "Hipertext.net." [Online]. Available: http://www.hipertext.net

[4] H. Kautz, B. Selman, and M. Shah, "Referral web: Combining social networks and collaborative filtering," *Communications of the ACM*, vol. 40, 1997.

[5] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, 2003.

[6] B. Heitmann, and C. Hayes, "Using Linked Data to Build Open, Collaborative Recommender Systems," In *Proc. AAAI'2010,* AAAI Press, 2010.

[7] N. Zlatareva, "Processing RDF Triplestores in Front-End Applications Based on Context-Dependent Rules," In *Proc. International Conference on Computational Science and Computational Intelligence (CSCI'2016),* 2016.

[8] L. dos Santos, R. Carvalho, M. Ladeira, L. Weigang, and G. Mendes, "PR-OWL 2 RL – A Language for Scalable Uncertainty Reasoning on the Semantic Web," In *Proc.11th International Workshop on Uncertainty Reasoning for the Semantic Web*, 2015.

[9] T. Lukasiewicz, M. Martinez, L. Predoiu, and G. Simari, "Probabilistic Ontological Data Exchange with Bayesian

Networks," In *Proc.11th International Workshop on Uncertainty Reasoning for the Semantic Web*, 2015.
[10] J. McCusker, M. Dumontier, R. Yan, S. He, J. Dordick, D. McGuinness, (2017), "Finding melanoma drugs through a probabilistic knowledge graph," *Peer J Computer Science* 3:e106. [Online]. Available: https://doi.org/10.7717/peerj-cs.106
[11] N. Zlatareva, "Context - dependent Reasoning for the Semantic Web," *Journal of Systemics, Cybernetics and Informatics*, vol. 9, no. 4, IIIS Press, 2011.
[12] N. Zlatareva, N. Swaim, M. Fitzgerald, S. Bagheri-Marani, and J. Watkins, "Building an Application Agent for the Semantic Web," *Proc. 9th International Multi-Conference on Complexity, Informatics and Cybernetics (IMCIC'18)*, March, Orlando, FL, 2018.
[13] D. Allemang and L. Hendler, *Semantic Web for the Working Ontologist*. Morgan Kaufmann, 2012.