Machine Learning and Text Analysis in the Tasks of Knowledge Graphs Refinement and Enrichment

Victor Telnov[0000-0003-0176-5016] and Yuri Korovin[0000-0002-8399-4439]

National Research Nuclear University "MEPhI", 249040 Obninsk, Russia
telnov@bk.ru

Abstract. Working prototypes of the scalable semantic web portals, which are deployed on cloud platforms and intended for use in universities educational activity, are discussed. The first project is related to teaching in the field of nuclear physics and nuclear power engineering. The second project is related to training in computer science and programming. The possibility of using the DL-Learner software in conjunction with the Apache Jena Reasoners in order to refine the ontologies that are designed on the basis of the SROIQ(D) description logic is shown. A software agent for the context-sensitive searching for new knowledge in the WWW has been developed as a toolkit for ontologies enrichment. The binary Pareto relation and Levenshtein metrics are used in order to evaluate the measure of compliance of the found content concerning a specific domain. It allows the knowledge engineer to calculate the measure of the proximity of an arbitrary network resource about classes and objects of specific knowledge graphs. The suggested software solutions are based on cloud computing using DBaaS and PaaS service models to ensure the scalability of data warehouses and network services. Examples of applying the software and technologies under discuss are given.

Keywords: Knowledge Database, Ontology Engineering, Context–Sensitive Search, Semantic Annotation, Cloud Computing, Education

1 Introduction

The world of data is a place where computers rule. Supercomputers have wondrous capabilities, but they often find it difficult when it comes to acquiring new knowledge and experience or existing knowledge categorization. While it's natural for a human to decide whether two or more things are related based on cognitive associations, a computer often fails to do it. The endowment of machines with common sense, as well as domain–specific knowledge in order to give them an understanding of certain problem domains, has been and remains the main goal of research in the field of artificial intelligence. While the amount of data on the WWW, as well as in corporative intranets headily grows, knowledge databases engineering still remains a challenge. This paper discusses, how the semi–automatic methods work for knowledge graphs refinement and enrichment.

A recent authoritative review of the latest achievements and current issues in the designated field of the Semantic Web is given in [1]. Our main practical contribution in this area is to develop working prototypes first, then scalable semantic web portals, which are deployed on cloud platforms and intended for use in universities educational activity. The first project [2] is related to teaching in the field of nuclear physics and nuclear power engineering. The second project [3] is related to training in computer science and programming. The potential recipients of solutions and technologies that are introduced in the projects mentioned above are students, professors, experts, engineers and handlers, which concentrate in the particularised domains.

2 Knowledge Graphs Refinement. Case Study

2.1 Knowledge Representation. Ontology Design

Ontologies are often considered as special knowledge repositories that can be read and recognised both by people and computers, separated from the developer and reused. Ontology in the context of information technology is a formal specification with a hierarchical structure, which is created to represent knowledge. Typically, an ontology holds descriptions of classes of entities (concepts) and their properties (roles) with respect to a certain subject domain of knowledge, as well as associations between entities and constraints on how these associations can be used. Further, we adhere to the formal definition of ontology, which is given in [4]. Ontologies, which additionally incorporate objects (instances of entity classes) and particular statements about these objects, are also referred to as knowledge bases or knowledge graphs.

The initial ontology design is performed by a knowledge engineer with the involvement of domain experts. In the process of creating a quality ontology, clearly articulated databases normalization principles should be taken into account that reflects best practice. Web ontologies at the design stage already define a hierarchical structure of knowledge, which can be expressed explicitly or it may be detected indirectly, based on the available axioms of categorization. During the subsequent practical use of the ontology, it may turn out that some designed classes are sparsely populated since the created hierarchy reflected the subjective point of view of the knowledge engineer (ontology designer), which does not correspond to the actual filling of the ontology with specific resources.

In this case, the reengineering the structure of the created ontology is inevitable. Adequate allocation of resources into appropriate classes is a fundamental intellectual service of the semantic web. Existing clustering methods offer an effective solution to support a variety of complex related actions, such as building ontologies, taking into consideration the inherent incompleteness underlying the representation of the facts. Among the large number of algorithms proposed in the machine learning literature, the concept cluster approach [5], which integrate the Dempster–Shafer theory with learning methods for terminological decision trees, and is designed to extract an authentic hierarchy based on actual resource allocations, deserves special attention.

There are two features that should be taken into account during the ontology design. The first feature is the Open World Assumption. OWL by default considers the

world open, that is, everything that is not explicitly specified in the ontology is not considered as false, but considered as possible. Facts and statements that are false or impossible should be clearly stated as so in the ontology.
The second feature concerns the use of the disjointness axioms in the ontology. Why is class disjointness important? This is due to the reliability of the logical entailments that can be obtained from the ontology. The point is that OWL does not imply class disjointness until it is explicitly declared. That is, by default, classes may overlap unless otherwise specified. If anyone simply declares two classes A and B in the ontology and say nothing more about them, than any ontology model can interpret these classes as it pleases, for example, as nested classes, as intersecting classes or as disjoint classes. However, if two entities in the domain really belong to different classes, this fact should be reflected in the ontology. One of the purposes of ontologies is to make domain knowledge explicit and reliable, which is why the axioms of disjointness are so important.

As an illustration of the ontology engineering process in the project [3], Fig. 1 below shows a design pattern for an ontology «Computer Science Training Center». This pattern was created on the basis of an analysis of the curriculums of the following Russian training centres: National Research Nuclear University MEPhI and Moscow State University, Faculty of Computational Mathematics and Cybernetics. The ontology design pattern is expressed in the UML notation according to the worldwide standard [6]. The actual ontology in serialized format (OWL2 XML syntax) is available at the reference [7].

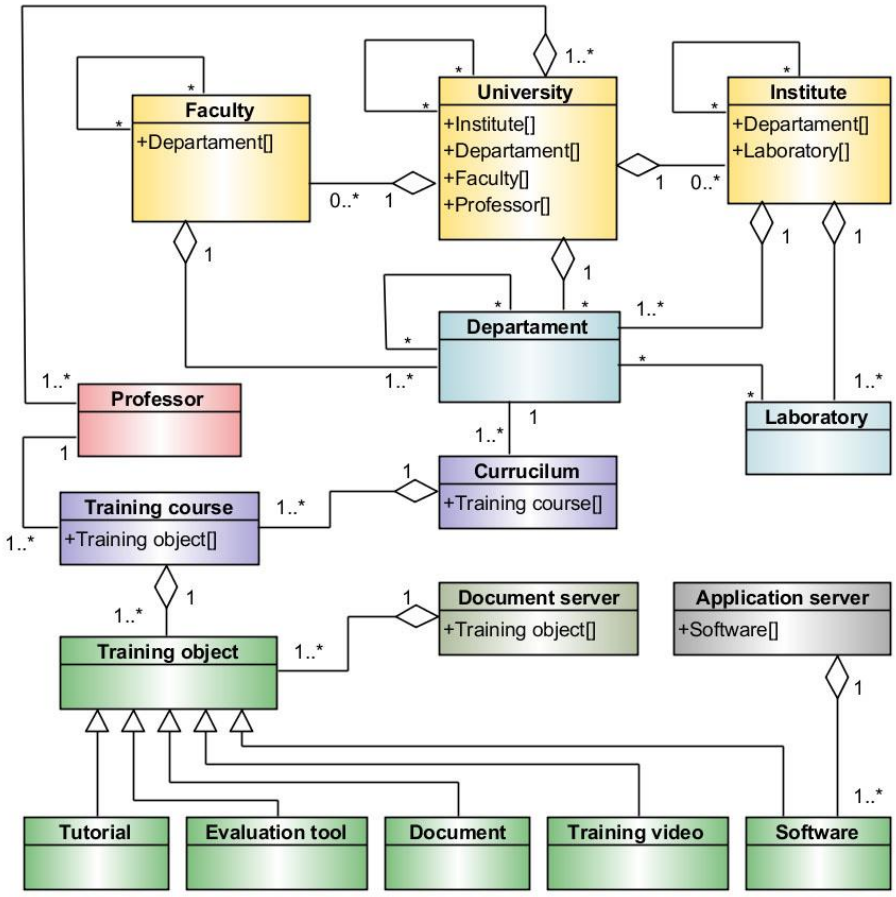

Fig. 1. The design pattern for an ontology «Computer Science Training Center» in UML notation.

2.2 Ontology Refinement

Fig. 2 below shows the ontology refinement process using machine learning methods; Protege plugin DL-Learner [8] works. This ontology is based on the pattern shown in Fig. 1. All top–level classes are disjoint. However, second–level classes, such as «Software» and «Tutorial», may overlap. This is due to the peculiarity of the domain, where some software components are both the tutorial and the working software.

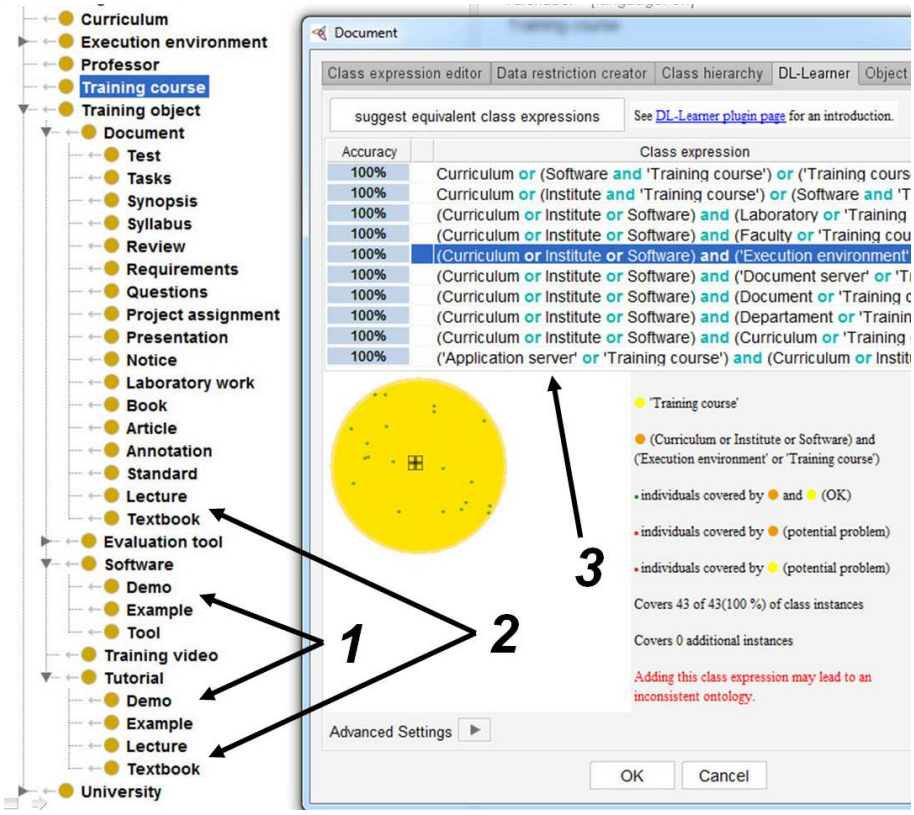

Fig. 2. Examples of disjointness violation into the ontology «Computer Science Training Center»: 1 – the classes «Demo» and «Example» are subclasses of «Software» and «Tutorial» simultaneously; 2 – the classes «Lecture» and «Textbook» are subclasses of «Document» and «Tutorial» simultaneously; 3 – axioms proposed by DL-Learner for ontology refinement.

The particularity of our case study was that when working with the DL-Learner, we did not use the built-in Pellet, FaCT ++, HermiT or OWLlink reasoners, since they are mainly focused on the use of ALC-level description logics. We investigated the ontology, which is designed on the basis of the more rich SROIQ(D) description logic, therefore the more advanced Apache Jena Reasoner to the DL-Learner [9] has been connected. A full description of the syntax and semantics of the used description logic is given in [4]. Specifically, in [4] predicates are listed and their interpretation is given in the subject area for the following constructs: atomic concept; abstract role; specific role; nominals; data type; conjunction; disjunction; negation; interpretation of the existential quantifier for concepts and roles; the interpretation of the universal quantifier for the concepts and roles; restrictions on the cardinality of the roles from above and below; interpretation of the existential quantifier for data types; the interpretation of the universal quantifier for data types; reflexivity and transitivity of the roles; belonging of an individual to a concept; application of the role to the individuals; equality and inequality of the individuals; the identity of the concepts; subsump-

tion of the concepts; the identity of the roles; subsumption of the roles; non-overlapping roles; compound axioms of the roles nesting.

Nowadays the DL-Learner software product is perhaps the most popular of the available solutions for semi–automated ontology engineering. To reveal dependencies hidden in the ontology, the refinement operators, heuristic measures and training algorithms named OCEL and CELOE are used, see [10]. It guarantees that the axioms proposed are minimal in the sense that one cannot remove parts of them without getting a non-equivalent expression. It makes use of existing background knowledge in ontology coverage checks. Statistical methods are used to improve the efficiency of the algorithms, such that they scale for large knowledge bases.

3 Knowledge Acquisition. Context–Sensitive Search

As a toolkit that provides data for ontologies refinement and enrichment, a software agent (which is actually a specialized meta–search engine) for the survey context–sensitive search for new knowledge in the WWW is implemented. To begin with it, should be noted several essential features of public search engines that are well known to most maximum users.

- the content found are ordered by the public search engine in accordance with its intrinsic algorithm, which does not always meet the interests of a special user;
- users are not always convenient to manage the context of the search query, clarify and focus the search;
- links to the business sites usually have a higher rank than other search results. Such effect is gained through the use of so-called search engine optimization (SEO) to artificially inflate the positions of commercial network content on pages of public search engines, in order to boost the flow of potential consumers for the subsequent monetization of traffic.

It appears that the above points and inclinations make known search engines an increasingly incompetent tool for extracting knowledge in the WWW for educative purposes. The context-sensitive search is based on a simple idea: to create such a mediator (a software agent) between the knowledge engineer and public search engines that help to arrange search results in accordance with his professional wants, by effectively sifting inappropriate content and trash. The aim is to include the power of the modern search engines in the maximum level, including built-in query languages and other search handles.

When the «Context-sensitive search» software agent is operating, the global content search, as well as the search on the specific web resources, is originally conducted by the conventional search engines (Google Ajax Search, Yandex, Yahoo, Mail.ru), the communication with which occurs asynchronously via the vibrant pool of the proxy servers, each of which is hosted on the Google Cloud Platform. The results of the activity of the conventional search engines are a kind of «raw material» for extra processing. Especially designed proxy servers on the cloud platform parse these results and generate the feeds, which are then forwarded to the client computer, where from the feeds the snippets are formed. These snippets, which metadata, before they arrive on the monitor of the client computer, undergo supplementary processing,

screening and sorting, as outlined below. In particular, for each snippet, its relevance, persistence and a number of other indexes are computed, which are then used to organise and clustering search results retrieved.

3.1 Search Context

The query language of some search engines may involve the so-called «search context». It is about using immediately in the text of the search query of particular operators, which allow the user to designate the presence and relative locating of specific tokens in the content found. In this paper, a «search context» is understood a slightly another way, namely, as a certain limited on length text that designates the domain that is currently of matter to the knowledge engineer.

When setting the search context, the subsequent data sources are available: taxonomies, thesauri, keywords, ontologies, textual files from the user computer, arbitrary content from the WWW. Any mixture of the above methods for setting the search context is enabled. The resulting context is the union of the chosen options. The context defined in this way allows us to choose, sort and classify information that originates from the search engines through the proxy servers. Fig. 3 below exposes the possible options for setting the search context.

Fig. 3. Establishing the context for the context-sensitive search: 1 – setting the context using a file from the client computer; 2 – setting the context using an arbitrary network content; 3 – widgets to show the established context.

3.2 Relevance, Pertinence, Metrics

For the goals of this paper, the relevance of the snippet is the measure of the similarity between the snippet and the text of the search query. Under the pertinence of the snippet is intended the measure of the similarity between the snippet and search context, that was determined earlier. These and other measures are estimated by means a fuzzy matching of the corresponding texts. To quantify these measures, «Context–

sensitive search» software agent uses the Levenshtein metrics [11]. Each lexical unit (token) from the snippet is sequentially compared with each token from the text of the search query. The algorithm for computing the snippet's pertinence looks alike, with the only difference that each token from the snippet is successively compared to each token from the search context. The process of assessing the relevance and pertinence of snippets is a formal one, without investigating the possible connections of individual tokens and their surroundings. It is believed that earlier such an investigation was performed to some extent during the primary search of the network documents and their full-text indexing in databases of traditional search engines.

Various options for classifying search results in the final output of the «Context-sensitive search» software agent is permitted. Deserves a particular mention the sorting by aspect named «dominance index», which supplies a joint account of the values of many metrics that describe the adequacy of the snippets. For example, the dominance index, in addition to the relevance and pertinence of the snippets, can also take into account the measure of the similarity between the snippet and the keywords, categories and attributes of the educational portal in total. For the practical calculation of the values of the dominance index, it seems reasonable to use the formalism of Pareto dominance relation [12], since Pareto's multi–criteria ranking does not presuppose an a priori knowledge of the relative importance of aspects (for example, what is more important, relevance or pertinence?).

Let given the original set of snippets, from which one should choose some optimal subset. The choice should be made on the basis of specific ideas about the adequacy of snippets (the principle of optimality). The choice task is a simple one if there is only a single aspect by which it is reasonable to compare any two snippets and directly indicate which one is more adequate. The solution to the simple choice tasks is naive. In real circumstances, it is not possible to single out any one aspect. Furthermore, it is often commonly hard to single out aspects. The selection and ranking of aspects that are quintessential for subsequent choice, in turn, is the task of choice. If some of the aspects are more significant (priority) of other aspects, this circumstance should be taken into account in the mathematical model of choice.

The selection task is the algebra $< \Omega, \mathrm{O} >$ where Ω is a set of alternatives (in our case, a set of snippets), and O is the optimality principle. The task makes sense if the set of alternatives is known. Usually the principle of optimality is unknown.

For further discussion, suppose that each snippet $x \in \Omega$ is characterized by a finite set of aspects $x = (x_1, x_2, ..., x_m)$. Let $\mathrm{A} = \{1, ..., m\}$ be the set of aspect numbers to consider when choosing; $\{\mathrm{A}\}$ is the set of all subsets A.

It can be assumed that choosing between any two snippets x and y with only one of any aspect taken into account is a simple task. If this is not the case, the corresponding aspect can be decomposed and presented as a group of simpler aspects. For each pair of snippets (x, y) we define a family of functions $\alpha_j(x, y)$ as follows:

$$\alpha_j(x, y) = \left\{ \begin{array}{l} 1, \textit{ if } x \textit{ exceed } y \textit{ in aspect } j \\ 0, \textit{ if } y \textit{ exceed } x \textit{ in aspect } j \end{array} \right\} \textit{ where } j \in \mathrm{A};\ x, y \in \Omega;$$

If x and y are equal or not comparable in some aspect with the number j, then for such number j the function $\alpha_j(x,y)$ is not defined. Let's form a set J of numbers of such aspects that x and y differ in these aspects

$$J = \{\ j : j \in \mathrm{A};\ \alpha_j(x,y)\ \textit{is defined}\ \},\ \ J \in \{\mathrm{A}\};$$

Next, we construct a metric that takes into account the number of aspects by which a particular snippet is inferior to all other snippets. Let there be two snippets $x, y \in \Omega$. Denote

$$d(y,x) = \sum_{j \in J} \alpha_j(y,x)$$

the number of aspects in which y is better than x. Then the value

$$D_\Omega(x) = \max_{y \in \Omega} d(y,x) \qquad (1)$$

is called the dominance index of x when presenting the Ω set. This value characterizes the number of aspects of the snippet x that are not the best in comparison with all other snippets available in the Ω set.

Let us define the function $C^D(\Omega)$ for selecting the best snippets as follows:

$$C^D(\Omega) = \{\ x \in \Omega :\ D_\Omega(x) = \min_{z \in \Omega} D_\Omega(z)\}$$

Here, the value $D_\Omega = \min_{x \in \Omega} D_\Omega(x)$ is called the index of dominance of the whole Ω set. Snippets with a minimum value of the dominance index form the Pareto set. The Pareto set includes snippets that are the best with respect to all the considered aspects, including relevance and pertinence.

In the projects [2, 3], an intuitively more acceptable value is used as the index of dominance, equal to the difference between the number of aspects taken into account and the dominance index determined by the formula (Eq. (1)). Accumulations of snippets with the same value of the dominance index form clusters, which in the final output of the «Context-sensitive search» software agent are arrayed in descending order of this index. As an instance of the previous statement, Fig. 4 below displays a variant of sorting snippets by dominance index. Snippets are arrayed in descending order of the dominance index value when six metrics are taken into account, including snippets relevance and pertinence. When snippets are arrayed by the value of the dominance index, within accumulations of elements with the same value of the dominance index (that is, within a cluster), the snippets are arrayed by each of the metrics taken into account in the computations.

4 Knowledge Graphs Enrichment. Semantic Annotation

Unlike conventional lexical search where search engines look for literal matches of the query words and their modifications, semantic annotation attempts to interpret ordinary language close to how people do it. During semantic annotation, all allusions

to cases related to entities in the ontology are appreciated. Semantic annotation is the adhesive that ties ontologies into document spaces, via metadata.

The workbench for executing the semantic annotation process is shown in Fig. 4 below. At the top of the workbench is a workspace for entering and editing network content addresses (URLs) to be annotated. The data in this workspace can be originated from any source, including manually. However, a more technologically high-level approach is to first obtain on the WWW those network resources that are most satisfying to a given domain using the «Context-sensitive search» software agent. The found suitable content can then be obviously loaded using the «Download resources» button and included in the list for annotation with a single mouse click.

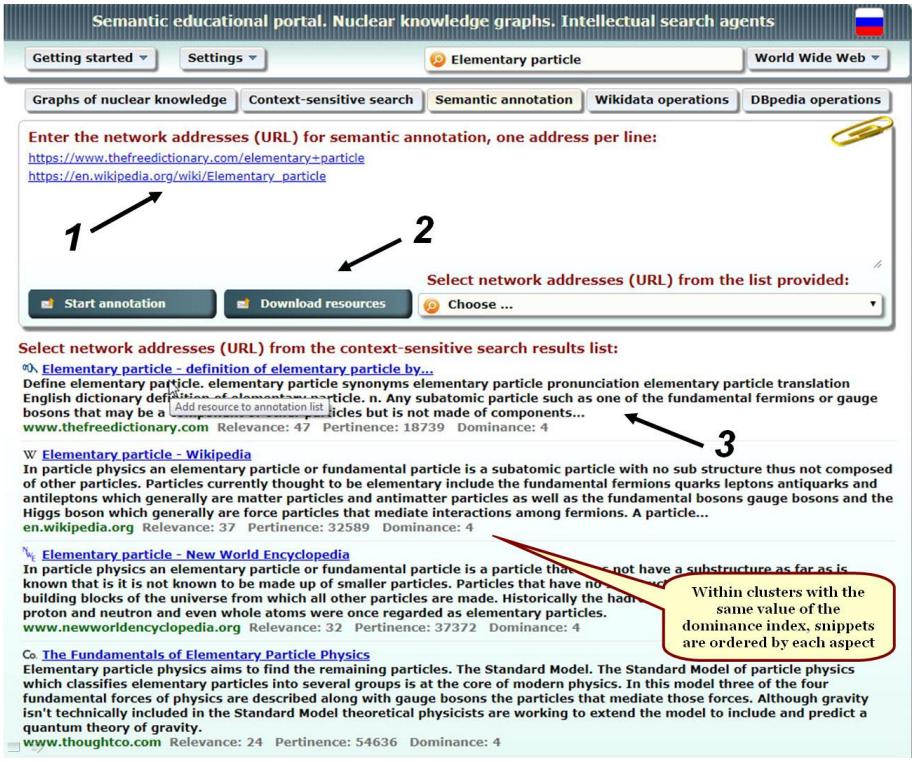

Fig. 4. Choosing network resources for semantic annotation: 1 – workspace for listing and editing network addresses (URLs) to be annotated; 2 – setting options, and loading results of the context-sensitive search; 3 – the most relevant results of the context-sensitive search.

The settings sheaf for the semantic annotation process is shown in Fig. 5 below. For annotation, you can select any of the knowledge graphs that are presented in the semantic repository, as well as any aggregate of them. To calculate measures of similarity between the annotated content and entities from knowledge graphs, both text analysis methods and neural networks that are trained on existing knowledge graphs can be practised.

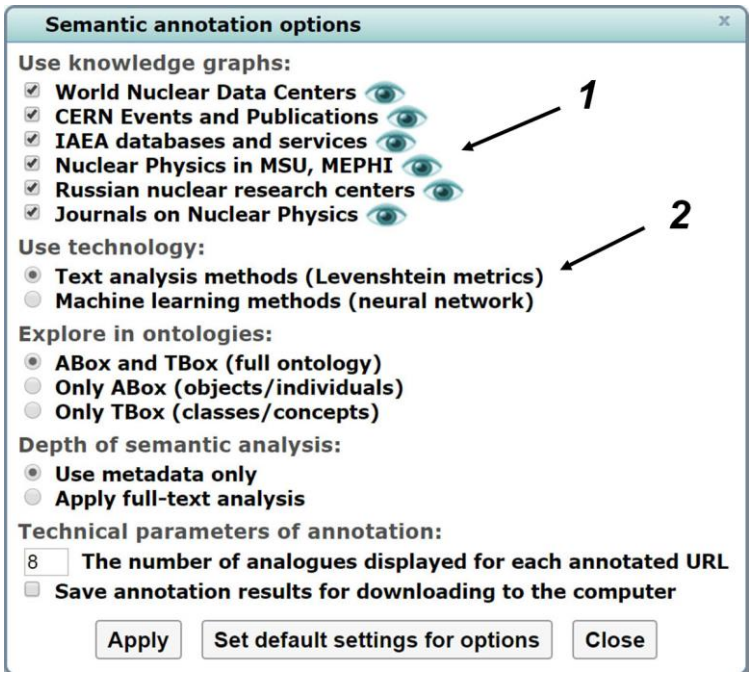

Fig. 5. Setting the options for the semantic annotation process: 1 – selecting and visualizing the knowledge graphs used; 2 – selecting of the technology and setting semantic annotation parameters.

It is possible to annotate network resources using classes (concepts) of the ontology (TBox - terminological components), using objects (individuals) of knowledge graphs (ABox - assertion components), or using both of them.

The depth of the semantic analysis can be limited by considering textual metadata inherent in network resources and entities in knowledge graphs. It can be very expensive to carry out full-text semantic analysis and in many ways redundant. Improving the accuracy of annotation in full–text analysis often does not justify the increased consumption of computing resources.

The number of entities from the knowledge graphs can be limited by the user. The entities that are most adequate to the annotated resource appear at the top of the output of the «Semantic annotation» software agent. All the results can be saved in files on the user's computer for later study.As an example of the use of a software agent "Semantic Annotation", Fig. 6 below shows the results of the semantic annotation of the network resource. It can be seen that have been discovered semantic annotation of five different graphs knowledge. With one click the user can open a browser and visualize RDF annotations found in any of the graphs of knowledge, as well as anyone can see the environment found entities, such as classes and their neighboring objects. This information is necessary for the knowledge engineering, which deals with refinement and enrichment of knowledge graphs..

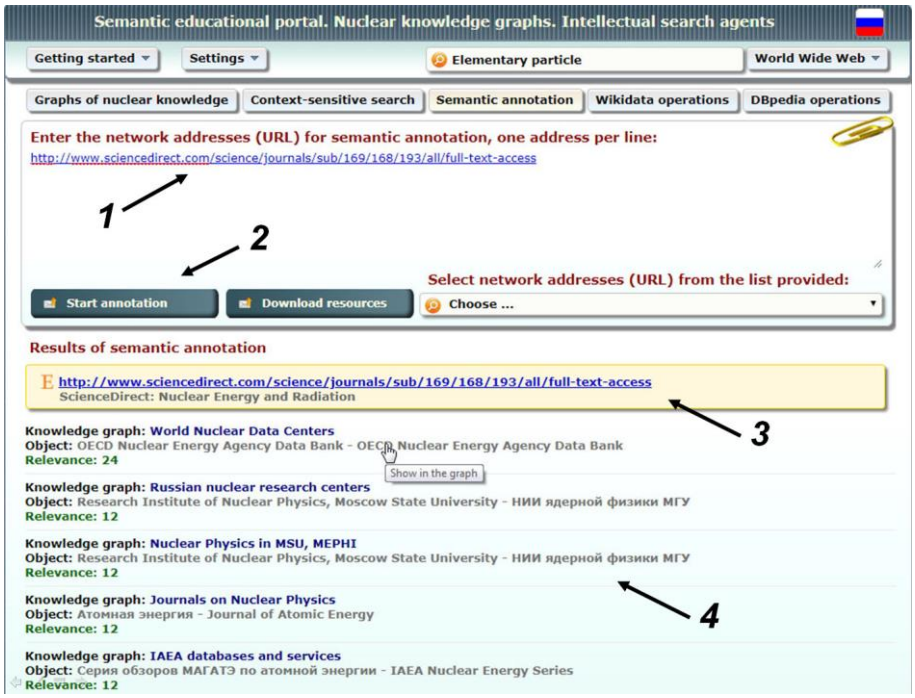

Fig. 6. Showing results semantic annotations: 1 - annotated addresses of network resources (url); 2 - settings and start the process of semantic annotations; 3 - network resource for which the semantic annotation is performed; 4 - knowledge graphs and essentially corresponding annotated network resource.

5 Related Work, Conclusion

Communities of scientists from the University of Manchester, Stanford University, the University of Bari, Leipzig University, Cambridge University and a number of other universities focus on the development of the theory and technology implementation for the Semantic Web, Description Logic and Machine Learning. Among the publicly available working frameworks that are designed to enrich knowledge graphs with content from the WWW, the REX project should be mentioned first [13]. Special mention goes to the project [14], where there was an attempt to put into practice the methods of inductive reasoning for the purpose of semantic annotation content from the WWW. Among modern industrial solutions aimed at corporate users, special attention should be paid to Ontotext Solutions [15]. The solution categorizes unstructured information by performing knowledge graph–powered semantic analysis over the full text of the documents and applying supervised machine learning and rules that automate classification decisions. This service also analyses the text, extracts concepts, identifies topics, keywords, and important relationships, and disambiguates similar entities. The resulting semantic fingerprint of the document comprises metada-

ta, aligned to a knowledge graph that serves as the foundation of all content management solutions.

It is worth commenting on the use of the Levenshtein distance to calculate a measure of similarity between two pieces of text. Today there are more than three dozen algorithms that solve similar problems in various ways [16]. For example, the Ratcliff-Obershelp metric [17] is based on finding matching substrings in tokens. When comparing two tokens, the simple, intuitive Ratcliff-Obershelp algorithm has an expected computational complexity of O(n*n), but O(n*n*n) in the worst case (where n is the length of the matched substrings). At the same time, the Levenshtein metric gives a similar result faster for a fixed computational complexity of the algorithm O(n*m) (where n and m are the lengths of the tokens being compared), and this algorithm is not recursive.

As for the use of the binary Pareto relation for multi-criteria ranking and clustering of the found network content, the use of this fruitful idea is not fundamentally innovative. For example, the Skyline software [18] has been actively using Pareto sets for working with databases for two decades. In our case, the software implementation of the Pareto optimality principle is peculiar, when a dynamically calculated dominance index allows us to categorize network content without storing it all in the computer's memory. Multi-criteria ranking of network content can be provided under the following conditions: 1) when groups of criteria are ordered by importance; 2) when the comparative importance is known only for some pairs of criteria; 3) when there is no information on the relative importance of the criteria.

It is necessary to develop and improve tools for the intuitive perception of linked data for non–professionals. VOWL [19] is one of the current projects for user-oriented ontology views, it offers a visual language, which is based on a set of graphics primitives and abstract color scheme. LinkDaViz [20] offers Web-implementation workflow that guides users through the process of creating visualizations by automatically classifying and binding data to imaging parameters. SynopsViz [21] is a tool for scalable multi-level plotting and visual exploration of very large RDF datasets and related data. The accepted hierarchical model provides an effective abstraction and generalization of information. In addition, it can effectively perform statistical calculation by using the aggregation hierarchy levels.

Unlike to the above decisions, the projects [2, 3] are mainly focused on the implementation of the educational activities of universities and are not limited to graph drawing knowledge and interactive navigation, and focus on the introduction in the educational process of the latest semantic web technologies, taking into account advances in an indefinite thinking. Both the results obtained and the software created are used in the real educational process of the National Research Nuclear University MEPhI, and the project as a whole is focused on the practical development of semantic web technologies by students and teachers.

Acknowledgments

The reported study was funded by the Russian Foundation for Basic Research and Government of the Kaluga Region according to the research project 19–47–400002.

References

1. d'Amato, C.: Machine Learning for the Semantic Web: Lessons learnt and next research directions. Semantic Web 11(5), 1–8 (2020) DOI: 10.3233/SW-200388.
2. Semantic educational portal. Nuclear knowledge graphs. Intelligent search agents, http://vt.obninsk.ru/x/, last accessed 2020/04/20.
3. Knowledge graphs on computer science. Intelligent search agents, http://vt.obninsk.ru/s/, last accessed 2020/04/20.
4. Telnov, V., Korovin, Y.: Semantic web and knowledge graphs as an educational technology of personnel training for nuclear power engineering. Nuclear Energy and Technology 5(3), 273-280 (2019) DOI: 10.3897/nucet.5.39226.
5. Rizzo, G., Fanizzi, N., d'Amato, C., Esposito, F.: Approximate classification with web ontologies through evidential terminological trees and forests. International Journal of Approximate Reasoning 92, 340-362 (2018) DOI: 10.1016/j.ijar.2017.10.019.
6. ISO/IEC 19505–2:2012(E) Information technology – Object Management Group Unified Modeling Language (OMG UML) – Part 2: Superstructure. ISO/IEC, Geneva (2012).
7. Ontology «Semantic web training course», http://vt.obninsk.ru/s/education-sw.owl, last accessed 2020/04/20.
8. DL-Learner, http://dl-learner.org/, last accessed 2020/04/20.
9. Apache Jena Reasoners, http://jena.apache.org/documentation/inference/#rules, last accessed 2020/08/03.
10. Lehmann, J., Fanizzi, N., Buhmann, L., d'Amato, C.: Concept Learning. In: Lehmann, J., Volker, J. (eds.) Perspectives on Ontology Learning, pp. 71–91. IOS Press, Berlin (2014) ISBN 978-1-61499-378-0, DOI: 10.3233/978-1-61499-379-7-i.
11. Levenshtein, V.: Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady 10 (8), 707–710 (1965).
12. Chen, Y., Wang, Z., Yang, E., Li, Y.: Pareto-optimality solution recommendation using a multi–objective artificial wolf–pack algorithm. In: Proceedings of 10th International Conference on Software, Knowledge, Information Management & Applications (SKIMA), pp. 116–121. Chengdu, China (2016) DOI: 10.1109/SKIMA.2016.7916207.
13. REX: Web-Scale Extension of RDF Knowledge Bases, http://aksw.org/Projects/REX.html, last accessed 2020/08/03.
14. d'Amato, C., Fanizzi, N., Fazzinga, B., Gottlob, G., Lukasiewicz, T.: Combining Semantic Web Search with the Power of Inductive Reasoning, http://ceur-ws.org/Vol-527/paper2.pdf, last accessed 2020/04/20.
15. Ontotext Solutions, http://www.ontotext.com/solutions/content-classification/, last accessed 2020/04/20.
16. TextDistance, http://pypi.org/project/textdistance/, last accessed 2020/08/03.
17. Ratcliff/Obershelp pattern recognition, http://xlinux.nist.gov/dads/HTML/ratcliffObershelp.html, last accessed 2020/08/03.
18. Kalyvas, C.. Tzouramanis, T.: A Survey of Skyline Query Processing, http://arxiv.org/ftp/arxiv/papers/1704/1704.01788.pdf, last accessed 2020/08/03.

19. Schlobach, S., Janowicz, K.: Visualizing ontologies with VOWL. Semantic Web 7, 399-419 (2016) DOI: 10.3233/SW-150200
20. Thellmann, K., Galkin, M., Orlandi, F., Auer, S.: LinkDaViz – Automatic Binding of Linked Data to Visualizations. In: Proceedings of the 15th International Semantic Web Conference pp. 147-162. Bethlehem PA USA (2015).
21. Bikakis, N., Skourla, M., Papastefanatos, G.: rdf:SynopsViz – A Framework for Hierarchical Linked Data Visual Exploration and Analysis. In: Proceedings of the European Semantic Web Conference ESWC pp. 292-297. Heraklion Crete Greece (2014).