Knowledge graph based distributed infrastructure for processing documents used for organizing education process

Evgeny A. Cherkashin[1,2], Victoria A. Popova[2]
[1]*Matrosov Institute for System Dynamics and Control Theory SB RAS, Irkutsk, Russia*
[2]*Institute of Mathematics and Information Technologies, Irkutsk State University, Irkutsk, Russia*
eugeneai@icc.ru; victorypopova1@gmail.com

Abstract:

Abstract:
This paper deals with the application of early developed infrastructural components based on knowledge graph data representation and knowledge-based data processing to university course documents processing, authoring and providing with their data a future platform for organization of the educational process. Main goal of the activity is to integrate static university facade data represented as educational program documentation with university infrastructure, e.g., library, schedule, various existing process planning systems already made by master students and faculty.

Keywords:
knowledge graph, logical inference, education process automation, distributed data processing, document authoring automation

1. Introduction

University is a complex sociotechnical system (STS) [1] comprising various components, which functioning is provided mostly by university staff and automation software. Irkutsk state university (ISU) has quite normal (required) level of automation in the areas of accounting, education process planning, learning management, student state control, library data access. The automation has an island character in domain and structural sense. Institutes of ISU develop own software to solve their actual problems and do not share results between ISU community. It seems that there is no such tradition. Some solutions are implemented by a special department of Institute of Mathematics and Information Technologies (IMIT) on request. For example, during COVID-19 pandemics, IMIT supported Big Blue Button functioning for all departments of ISU, implemented Moodle modules for remote enrollment management.

To obey constantly rising requirements imposed on the university functioning, some areas are still experiencing a lack of automation. One of the challenging problem is course documentation preparation, such as course description mediation with an academic plan (curriculum). In a course description (CD), in the case of contracting or changing the numbers of units (hours) spent to lecture and laboratory works, teacher must reschedule the amounts by adding/removing topics or contracting/extending a topic content. The format of printing layout of the CD is altered every two years in Russia, and even unchanged content must be reconstructed. Yet another task is to check the capabilities of the university library to supply printing editions for references of the CD, renew URL references to the electronic ones. Another problem deals with educational process management. There is no general system of class schedule planning and browsing, monitoring student progress, namely, the fact of attending a class and the obtained grades.

The above mentioned problems are solved by faculty manually using office automation software (Microsoft and LibreOffice), institute management staff develop recommendations and template documents to simplify and inspire faculty to proceed with authoring of the CDs in time and result in a good quality documents (in all aspects). The devised manually, class schedules are published on ISU

site as soon as they formed. The human factor is exposed in a degradation of the quality of requirements compliance. This is, partially, due to the present underestimation of the role of a teacher and economic reasons: teachers often have positions in other institutions and mentally not interesting in placing this activity to the first place in their everyday schedule.

It seems that the problems can be solved by using artificial intelligence at a good level. ISU site includes units for supplying students and faculty with the educational process documents, including curricula and CDs for the past few years for all study programs. Documents are represented in PDF format and can be informative sources of meaningful data, as well as various documents from Russian government sites containing references books for specialties, job requirements, *etc*.

The aim of the present research presented in the paper is to automate these creative activities of the faculty authoring CD, organizing processes, monitoring and control students' progress, and in the future, form a basis of educational process modeling to support the compliance checking, individual education trajectories of students.

2. Used development platform

The basis of our distributed warehousing is being driven by present semantic data servers, knowledge graphs (KGs) [2], such as Virtouso [3], ClioPatria [4], and by adapting rational databases to Semantic Web (SW) technologies [5] (Figure 1). The usage of the distributed KGs allows us developing subsystems independently, having in mind their facilities of coupling distributed data and services within common domains in a distributed network. Using KG as data structures enables a developer to create software complexes as interacting agents via KGs' content, reaching loose coupling between them. Another advantage of SW and KG usage is nowadays domain standardized descriptions represented as well-known vocabularies.

The kernel module of applications is KG server (B2) built on the base of Virtuoso with web-server and Pengines plug-ins [6]. Virtuoso supports UPDATE and DELETE SPARQL queries, can be scaled in a network. Pengines service enables SW web-applications to utilize logical inference on the server-side with browser JavaScript environment integration. Pengine's knowledge base is extended with the programmer's Prolog modules [7, 8]. The kernel module stores a common global part of A- and T-boxes of applications, *i.e.*, it is an application database. Other modules provide an application (C3) that use other platform assets.

Authoring tool (B1) [9, 10] is used to provide document generation, and integration. This tool allows one to implement SW document content generation utilizing templates and data from other web pages and websites providing LOD[1] [11, 12] capabilities.

Text indexing engine (C1) provides a service for storing documents represented as KG triples, JSONs, and BLOBS of any format with corresponding text layers. This module has two implementations. One is built on the Sphinx Search engine, and the second is Elasticsearch [13]. Elasticsearch supports JSON as the only document format and is easily integrated with KG using JSON-LD for triple representation. Sphinx Search is much faster in text and database records indexing and consumes less RAM as it is implemented in C++.

BLOB stored as KGs documents having no markup (PDFs, scanned documents) usually contain valuable data to be revealed for application use. The documents could be report data or scientific paper content, DJVU files, or raster images. Data recognition is implemented in the PDF/text document processing module (C2), where data processing workers query the text indexing module for BLOB documents and add recognized text and high-order structure layers. All obtained layers are stored in the database of C1, and data of general interest are to be converted in triples of KG.

1 Abbreviation of Linked Open Data

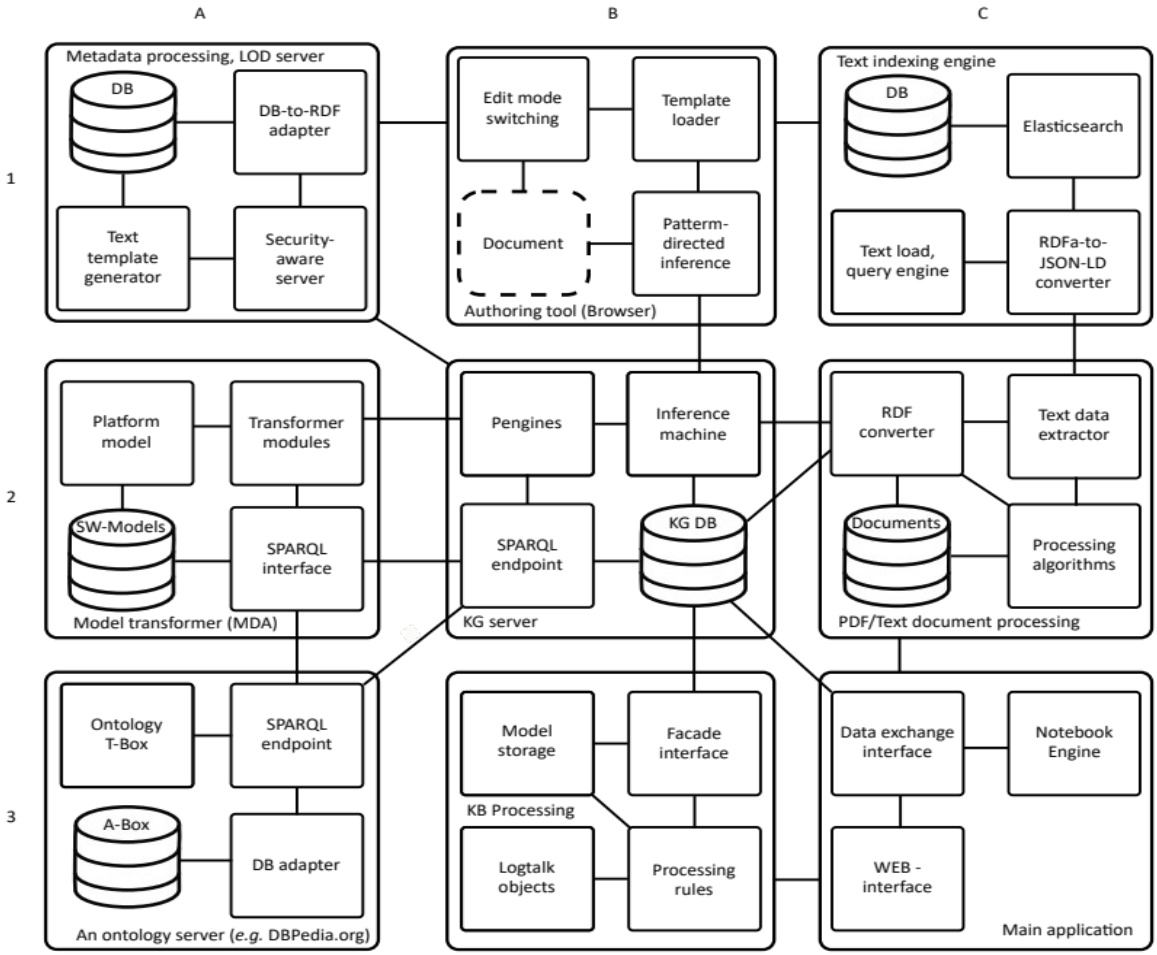

Figure 1: General architecture of used KG-based software development platform

A generalized KG processing is being executed in the KB processing module (B3). This component denotes a set of rules used to implement the validating, emergent semantics derivation, and analysis and synthesis of the new data, including resultant output. Module A2 is the source of model data (design of domains, UML, problem statement, *etc.*), which is converted and form the T-boxes of a KG. The conversion is a kind of Model-driven architecture (MDA) implementation [14].

The metadata processing module (A1) denotes the environment capabilities for expressing the semantics out of the data processing modules' output and the input data semantics. It allows us to store resultant data in KG for further usage in module (B3) for problem-solving and decision-making. Module A3 denotes external services and KG with valuable resources, *e.g.*, DBPedia [15] global objects.

3. Processing course description documents

As it was told above, the input consist of PDF documents of CD downloaded from the website of ISU. The document should first be analyzed to obtain meaningful data, which can be processed imperatively. It is usually possible as the PDFs are generated from MS Word documents (not scanned). For a CD, meaningful information is the title and the code of the course, list of topics, distribution of study units (academic hours) between lectures, practice, seminars, personal work of student, list of questions for knowledge assessment, *etc*.

The PDF processing is carried out by converting source PDF to XML by Poppler library. The XML file is processed by authors' set of utilities implemented as knowledge-based systems in

Logtalk [16] object-oriented (OO) logical programming language. Preliminary input data conversion represents XML tree (list of pages with lists of runs, lines and font definitions) into an in-object database, each element is numbered to conserve orders, neighborhood elements of the similar quality are marked to organize fast access to successor elements.

The system recognizes basic features of each run (a sequence of characters having a common style) and line, *e.g.*, is it a hanging line of a paragraph, or does it have a number at the start of its text. For all text lines of a page, we figure out the bounding box, excluding page numbers (one or two character strings at upper or lower edge of page with number that equal to the PDF page number). These features are used to join runs and rows into paragraphs, the bounding box is also used to figure out regular paragraph lines. Each join is accompanied of figuring out paragraph geometry. Paragraphs, which were separated by page breaks, are reconstructed at the second stage, using additional rules.

The next stage is CD structure recognition. Each institute of ISU has templates for CD authoring, so that the sequences and styles of the headers are similar but varying between institutions. The headers are recognized by their numbers in a template, verifying a sequence of roots of comprising words. The template variations are accounted by specifying a categories of special rules, adjusting parameters of general rules and adding special ones. Then all lines/paragraphs are associated to the header preceding them. The headers and subheaders are in hierarchical relations too. So, at this stage, we have a tree of basic document structures.

After constructing the hierarchy, the process of list recognition is launched. We suppose that bullet lists have a deeper level of folding as compared to numbered ones. The process has two stages: running through all list-like structures and find common sequences of constantly increasing numbers or same bullet symbols trying to conserve possible folding, and folding itself. If a one-line list is found, this case is considered as a false positive list item, and it is joined to the previous paragraph as a regular line.

The last stage is meaningful content acquisition. At this stage, the data about header contexts and list folding are significant, they constrained the set of paragraphs, where meaningful data are located. A location is specified with the context and a list of preceding word roots and punctuation. The target data are extracted with regular expressions or other string processing procedures.

The main OO programming technique used in implementation of PDF documents analysis is object extension and composition. The XML-data of a document is encapsulated as a parametric Logtalk object [16], which parameter is its path name. The object provides basic input-output functions, conversion from a tree to the database, database content modification conserving consistency, debug printing, and so on. It is extended to a recognition object (RO) by importing a set of categories implementing analysis and synthesis stages, a set of configuration predicates are added as well. RO is constructed aimed at a special case of CD template, so the imported categories are extended by descendants improving recognition capabilities. As compared to the present component frameworks, configuration is realized with the same programming structures as a predicate implementation thanks to the abstract syntax of Prolog/Logtalk, and, moreover, configuration parameters also can be rules deducing values. The extracted data are converted to a local KG, which added to the remote KG storing all CD data.

3.1. Authoring and generating a course description

CD data stored in a KG are filled in LuaLaTeX template based on a special LaTeX class `sucourse`, a subclass of a KOMA-script `scrartcl`, having special commands and environments for defining distributions of study units for various educational activities via keywords.

For example, the laboratory works are defined with two environments `labworks` that setups an enumeration environment and an RDF context for SW data acquisition, and `work` defining a concrete problem definition. The functionality of environments used is defined with so-called new LaTeX extended syntax, allowing the programmer to control types of parameters. The current alpha version of class `sucourse` is published at Github [17].

The environments implicitly execute Lua code for collecting data, checking constraints and generating tables and other TeX structures of a CD. If constraints are not satisfied, an error message is added as red color text into the TeX source. The validation can be turned off by final option to class. Lua's code also generates auxiliary files with data that are to be processed between LuaLaTeX tool executions.

The latter feature is used to check library capability to support printed editions, and renew URL references for electronic ones. This service is implemented asynchronously using RabbitMQ server to organize workflows in LAN. One of KG subgraphs stores editions' data (caching), and texts of the references are substituted. Other collected data from LaTeX markup of CDs is sent to KG as well.

The usage of LuaLaTeX as the main text processing unit is dictated by its outstanding text layout quality capabilities, high-level markup by new commands and environments, true type font availability, and Lua support as an extension language allowing one to communicate with a parallel process in a convenient way. Special commands prevent user to change styles, also by defining useful new structures we can support development the documents with a common text structures representing various aspects of a CD. A typical example is tests, which can be integrated in a document and then be exported into a format understood by LMS Moodle.

Initial state of a LuaLaTeX CD representation is generated by Python and an RDF extended Jinja templating engine. Jinja was improved with syntax structures allowing one to query KG or its local subgraph for triple data. There are value-like queries with rdf:type restrictions, and environments for answer set processing and a context definition. Results of a template rendering also include a semantic markup invisible in the terminal PDF, but allowing one to untangle RDF data from LuaLaTeX CD sources (alike RDFa for HTML).

4. Scheduling

Devising of the schedule is a well-known problem. ISU requirements to the complexity of the solution are rater soft. This is because many requests for the schedule are difficult to formalize, and they are already considered during manual editing. The schedule currently is not compiled centralized, but for each institute or faculty separately. To form a schedule of a good quality (satisfying faculty and students' groups as well as possible, minimizing class resources), it is necessary to take into account a number of constraints, such as a ban on free time intervals instead of classes, control of the capacity of classrooms, various requests of faculty, for example, which days are more convenient to conduct classes, the maximum number of classes per day, *etc*. Although it is not always possible to account all the requests for the schedule, the ISU requires a system that could make a preliminary version of the schedule with the possibility of a convenient manual editing.

To devise a schedule, the constraints must be presented in a structured format that the system is capable to load and process. To form the constraints' set directing the scheduling, it is necessary to use the following source data:

- for classroom: the number of seats, the availability of a projector, computers, *etc*.;
- for faculty (preference and requests):
  - number of working days per week;
  - maximum number of classes per day;
  - the desired schedule of classes during the school week (compact, evenly, no preference);

- for student groups:
  - academic load (the total number of academic hours in each discipline, as well as the number of classes per week);
  - number of students in groups;
  - features of classes (streams, electives, *etc*);
  - last name, first name and patronymic of the teacher for each pair;
  - desired classroom for specific course class.

The new scheduling program version must load collected data from KG. To achieve these requirements, one is to construct a knowledge-based system inferring data from KG contents. The main source of the data are the curricula, faculty assignments to the courses, which also must be implemented, and institute resource descriptions, which have also somewhat dynamic nature.

5. Related implementations in other high schools

During the search of analogues and working in collaboration with other high schools of Irkutsk and Saint-Petersburg, we collected data about their software experience, which will be projected to our requirements.

Saint-Petersburg Electrotechnical University (ETU) developed a system for study program development automation [18]. The system enables faculty entering the meaningful information in to a JSON-base storage by filling in forms. Each document is represented as a JSON-structure. CDs are generated into LaTeX sources and processed into PDF. Chair secretary obtains the PDFs and publish them a course home page. The system is constantly developed in a department. At present, faculty member roles and document flow is realized, *e.g.*, the documents are verified by a role dealing with standard compliance. The state of a document is presented in a user interface. The system is capable to store arbitrary PDF documents and generate whole package of documents to present qualification authorities.

Less improved generator is developed in National Research State Technical University (NRTU) [19]. A server-based PHP application is loaded with curricula from well-known in Russia program “Shakhty”, presents a user interface to define formal parts of the CD, such as lecture topics, list of laboratory works, personal works and seminars with the distribution of study units between them. At the final stage, a Word document in `docx` format is generated. It should be filled in manually with other data, namely, list of tests, exams questions, literature references, *etc*.

The similar principle of helping teachers is in the base of Lan’ book publisher [20], but their system is oriented on an automation of electronic editions collecting for a CD’s syllabus. The information system stores all the books structuring by domain tag set and delivers content to the user filtered by similar topics regarding the set of the previous chosen exemplars.

Googling the Russian part of the Internet results in more examples of CD generators, quick review of their user guides reveals a common direction of the development to improve generation capabilities. In our R&D we are to construct a system which uses the existing structured and unstructured data to construct an information model of an education process of an ISU institute and scale the results to other departments.

6. Conclusions

At present stage of the research, we implemented a set of services on the base of a developed Knowledge Graph (Semantic Web) based infrastructure. The infrastructure’s modules’ processing is based on active use of metadata, distributed and remote semantic resources. It enables us to construct our solution in a multiagent fashion, where agents are independent of each other, provide a common domain for data structure devising.

Among the quite successive direction of our development are two subprojects we told about in this paper:

- course description data structure recognition from PDF documents, and
- their generation using a new curriculum and the recognized data.

Further development will be carried on in two main directions;

- improving quality of recognition with software processing tables from PDF documents using open software of our colleagues [21, 22],
- implementing a schedule compiling software with input data inferred from KG collected data.

Acknowledgment:

The results were obtained within the state assignment of the Ministry of Education and Science of Russia, the project "Methods and technologies of a cloud-based service-oriented digital platform for collecting, storing and processing large volumes of multi-format interdisciplinary data and knowledge based on the use of artificial intelligence, a model-driven approach, and machine learning", No. FWEW-2021-0005 (State registration No. 121030500071-2), the network infrastructure of the Telecommunication center of collective use "Integrated information-computational network of Irkutsk scientific-educational complex" (http://net.icc.ru) was used as well.

References:

[1] Z. Stojanov, J. Stojanov, G. Jotanovic, D. Dobrilovic. Weighted networks in socio-technical systems: Concepts and challenges. CEUR-WS Proceedings of the 2nd International Workshop on Information, Computation, and Control Systems for Distributed Environments Irkutsk, Russia, July 6-7, 2020. p. 265–276.

[2] A. Hogan, E. Blomqvist, M. Cochez, C. D'Amato. *et al*. Knowledge Graphs – 2020 URL:https://arxiv.org/abs/2003.02320v5 (access date: 12-Dec-2021)

[3] O. Erling. Virtuoso, a Hybrid RDBMS/Graph Column Store // IEEE Data Eng. Bull. 2012. Vol. 35 – pp. 3–8.

[4] J. Wielemaker, W. Beek, M. Hildebrand, J. Ossenbruggen, ClioPatria: A SWI-Prolog infrastructure for the Semantic Web, Semantic Web. Vol. 7(5). P. 529–541 (2016)

[5] T. Berners-Lee, J. Hendler, O. Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American, May 2001.

[6] T. Lager, J. Wielemaker. Pengines: Web Logic Programming Made Easy. Theory and Practice of Logic Programming, volume 14, No. 4-5, 2014, pp. 539–552.

[7] J. Wielemaker, G. Schreiber, B. Wielinga, Prolog-based infrastructure for RDF: scalability and performance, In: D. Fensel, K. Sycara, J. Mylopoulos (eds) The Semantic Web – ISWC 2003. ISWC 2003. Lecture Notes in Computer Science. Vol. 2870. Springer, Berlin, Heidelberg, 2003.

[8] J. Wielemaker, T. Schrijvers, M. Triska, T. Lager. SWI-Prolog. Theory and Practice of Logic Programming, volume 12, No. 1-2, pp. 67–96, 2011, ISSN 1471-0684.

[9] E. Cherkashin, A. Shigarov, V. Paramonov, A. Mikhailov, Digital archives supporting document content inference, Procs. of 42-nd International Convention on Information and Communication Technology Electronics and Microelectronics (MIPRO), May 20–24, 2019. pp. 1037-1042.

[10] E. Cherkashin, A. Shigarov, V. Paramonov. Representation of MDA transformation with logical objects // Procs. of International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON) Novosibirsk, Russia – 2019 – pp. 0913–0918

[11] Ch. Bizer, N. Heath, T. Berners-Lee, Linked data – the story so far, International Journal on Semantic Web and Information Systems. 2009. Vol. 5 (3). P. 1–22.

[12] N. Heino, S. Tramp, N. Heino, S. Auer, Managing web content using linked data principles – combining semantic structure with dynamic content syndication, Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual. pp. 245 - 250. http://svn.aksw.org/papers/2011/COMPSAC_lod2.eu/public.pdf

[13] R. Kuć, M. Rogoziński, Mastering Elasticsearch - Second edition, Packet Publishing. 372 p. (2015)

[14] E. Cherkashin, I. Terehin, V. Paramonov. New transformation approach for Model Driven Architecture // Proceedings of the 35th International Convention MIPRO, Opatija – 2012 – pp. 1082-1087.
[15] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, *et al*, DBpedia – a large-scale, multilingual knowledge base extracted from Wikipedia, Semantic Web Journal. Vol. 6, No. 2, P. 167–195, IOS Press (2015)
[16] P. Moura. Programming Patterns for Logtalk Parametric Objects // In: S. Abreu, D. Seipel (eds) Applications of Declarative Programming and Knowledge Management. INAP 2009. – Lecture Notes in Computer Science – Vol. 6547 – Springer, Berlin, Heidelberg – 2011.
[17] A LuaLaTeX class for authoring course descriptions, URL:https://github.com/eugeneai/sucourse (access date: 10.10.2022)
[18] ETU "LETI", URL:https://etu.ru/en/university/ (access date: 10.10.2022)
[19] INRTU is a university with the best traditions…, URL:https://eng.istu.edu/ (access date: 10.10.2022)
[20] LMS Lan', URL: https://e.lanbook.com/(in Russian) (access date: 10.10.2022)
[21] A. Shigarov, V. Paramonov, P. Belykh, A. Bondarev, Rule-based canonicalization of arbitrary tables in spreadsheets, In: G. Dregvaite, R. Damasevicius (eds) Information and Software Technologies. ICIST 2016. A. Shigarov, A. Mikhailov, "Rule-based spreadsheet data transformation from arbitrary to relational tables," Information Systems, 71, (2017).