Знания-Онтологии-Теории (ЗОНТ-17)

Linked Open Data technologies for authoring and publishing text documents

Cherkashin E. A. 1,2,3, Shigarov A. O. 1,3, 2 Orlova I. V., Mikhailov I. S. 4

1 V.M. Matrosov's institute of system dynamics and control theory SB RAS   
 Lermontov str. 134, Irkutsk, 664033, Russia  
2National research Irkutsk state technical university, Lermontov str. 83,  
 Irkutsk, 664033, Russia  
3Иркутский scientific center SB RAS, Lermontov str. 134, Irkutsk, 664033, Russia  
4Limnological Institute SB RAS, Ulan-Bator str. 3, Irkutsk, 664033, Russia

[eugeneai@icc.ru, shigarov@icc.ru](mailto:eugeneai@icc.ru)

**Abstract. The problems of automating the creation of text documents with the active use of declarative tools and technologies of Linked Open Data is discussed. Software tools to compose HTML5 documents from various sources and present the document in a specified layout, such as a template or form, is being developed. The sources are web pages and the results of text processing on the server or client side (web browser) processing results. The technologies are based on the active use RDFa markup. Several examples of the application of the developed software are considered.**

Keywords: Linked Open Data, automated composing a document, semantic markup of document

1. Introduction

Technology Linked Open Data (LOD) [1] has been suggested by W3C consortium to represent the semantic information in the data published in a way that provides not only the possibility of its processing with software agents (Semantic Web), but also to link all available information into a single semantic graph using relations and global universal identifiers (URIs) of theory resources. The descriptive capabilities of semantic web technologies, HTML5 document publishing tools and LOD technologies form a infrastructural basis of authoring and publishing documents. The document is constructed out of individual parts (text and images) that are loaded from other servers with links to relevant resources. Resources represent both the static content and the results of an algorithm execution generating text content from source data. The LOD provide a logical markup for the information presented in a document, informative basis for the different variants of visual representation and interpretation, logical connections with other documents, export information into other documents, procedural processing, etc. An important advantage of LOD usage in creation of information environments is a weakening of the requirements to published information warehouses: the document itself is the formalized data warehouse. In some extent, this allows redirecting time spent on designing the database structure for a storage of partially formalized documents to the process of solving a substantive problem: the user (developer) markups the text data with semantic meaning.

The aim of the study is to design an approach to the development of software performing the functions of the authoring and typesetting of documents marked up according to LOD principles. To achieve the goal a set of server functions and client (web-browser) JavaScript routines has been developed that allows the user to create documents with the web browser. The functions support uploading of text fragments from other documents and their generation with server and client scripts. The resulting document is placed in the data warehouse, printed or downloaded to the user workstation as an HTML file. JavaScript scrips implement various document views, e.g., as input forms, conversion of parts of text, e.g., to lean nouns and substitute pronouns, generate content, etc.

The created editing tools are the basis of developed digital archives of documents intended for solving problems of creating documentation of university courses, representing research results of DNA/RNA sequencing. The use of LOD data formats, HTML processing tools of web browser and developed technologies allows one to solve a wide class of problems of formation of documentation, ranging from the formation of a meaningful part of the texts through the presentation of stylistic characteristics of the texts and the integration of logical markup into a global data access services.

The development of the tools for document representation on the basis of LOD aimed at the formation of the prototype of a global environment that supports automation of data transfer between documents of several companies, and create, ideally, a distributed document processing with the principles of social networks. The task is relevant, especially in a small business environments with no well-established information flows and relationships between documents. This is mainly due to the dynamism [and diversity] of the small business. Use of large-scale workflow automation software in the environments of such kind requires large financial expenses for the purchase and/or continuous improvement of the software product. Moreover, in companies with a deployed document flow automation, there is almost always a wide class of documents that are not integrated in the main document flow (e.g., memos, statements), documents with difficult-to-formalize information (e.g., job descriptions) and the documents received by the organizations from the outside having no contain logical markup (e.g., industry standards).

This work continues the research outlined in [2] in implementing client services operating on the platform of the web browser.

1. Technology of document authoring based on the LOD

The developed software for typesetting of text documents are based on technologies, which implement the HTML5 standard available in modern web browsers. The document is a web page that is composed out of different parts by means of the JavaScript routines functioning within the web browser. To make changes in the content of the page, built-in browser editing tools of web pages (contentEditable attribute) are used. The tools are managed by the special libraries (Medium Editor, https://yabwe.github.io/medium-editor/), which extend the basic functions of the built-in editor with document structure conversion and LOD markup.

The text of the document is being marked up with RDFa attributes. The semantic markup is the logical structure of the document, the domain relations between the entities presented in the document, for example, the distinct [parties] are described in an agreement, subject of agreement; formal data; the “part-whole” relationships sectioning the text document. In addition to the semantic markup, command structures are embedded in the text of a document. Such structures are used for algorithm activation of content conversion, for example, some structures activate the inclusion of parts of other document in the current one, other instruction organize grammatical transformations of sentences. The command structures are recognized more easily than the structures of semantic markup. The command structures interpretation mainly aimed at algorithmic text processing.

As the basis for the semantic markup of the documents, the results of the open project Dokieli [3] is used. The project focuses on implementation of HTML5 text-editing tools for authoring a scientific publication by several co-authors. Dokieli allow to mark up text with tags RDFa, to copy a document in its own repository and make an annotation of the text with readers' and authors' comments. In the markup of a document, the following ontologies are used.

**Open Annotation (oa). The ontology accepted in 2017, its main purpose is to describe the content (annotation) describing other content. Browser bookmarks are examples of such content. An oa resource (an bookmark) is in two major relations to other resources: oa:hasTarget, oa:hasBody. The first relation refers to the annotated content with a corresponding resource URIs, and the second one, which can be multiple, refers to an annotation resource, for example, a text description of the annotated resource or an associated comment. The edited document is an annotation to the target document, i.e., at the beginning the documents annotates itself, and after the completion of the final version of the document, the annotation annotates the generated immutable copy, e.g., the PDF file of the document.**

**Friend-of-a-friend (foaf) describes information about agents: physical and legal entities, together with software agents. This ontology is widely used to represent relations between entities (agents) in a social network.**

**Provenence (prov) provides a vocabulary of terms and relations describing the origin of the information. This ontology is a convenient way to refer the document's source parts originated from other documents. The prov ontology is a basis of informational flows descriptions in documents and their mutual relationships.**

**Dublin Core (dc) is used to describe the elements of authored annotations, for example, section headings, document title, document types, various descriptions contained in the document. In contrast to oa this ontology describes the documents mainly at metainformation level.**

**DBPedia resource (dbr) is the namespace of the objects (resources) of Wikipedia. Ontology dbr is a convenient way to refer to specific instances of objects and concrete classes, for example, the passport of a person (dbr:Passport), a city, etc.**

**Schema.org (schema) represents the objects that are recognized by the site scanning agents of Google, Yandex, Yahoo etc. This ontology is used in the case if none of the above specialized ontological have a relevant term or a relationship denotation. This ontology contains useful relationships like “part-whole” (schema:hasPart), “result of a creative activity” (schema:CreativeWork). The relations are used to describe the sectioning of the document, i.e., its hierarchical structure.**

In addition to the above-mentioned ontologies, the standard ontologies (RDF, RDFs, RDFa, XSD) and ontologies from the NEPOMUK project (https://userbase.kde.org/Nepomuk) describing objects stored in the index of full-text digital archives are used as well. They allow one to describe meta-information about the creative work of various kind in details, e.g., file, image, document, document section; and specify the type of document, text content, identification of works, etc. In our research, there was a labor-intensive stage devoted to the search ontologies to represent documents by subject areas. The resource Linked Open Vocabularies (http://lov.okfn.org/dataset/lov/) happened to be very useful reference book as well as support of the community of researchers, which develop this resource.

The architecture of the system is presented in Fig. 1 (all relations shown in the figure are bi-directional). The page is downloaded from a server where it is composed from a content stored in a database or a file system, results of an algorithmic data conversion of other sources, and elements of a template that includes the user view interface elements and interpretation modules on the displayed page. In the formation of the final form of the document, interpretation modules request the required data and resources on the source server or other Internet servers.

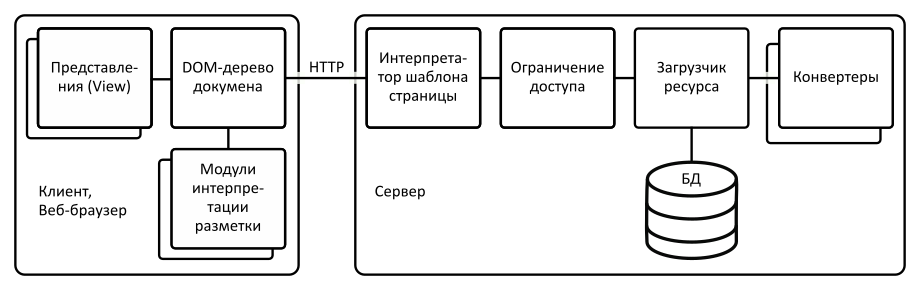


Fig. 1. The general architecture of the document publishing system

Interpretation modules of the semantic markup are implemented with client-side JavaScript. They are executed as soon as the main content is loaded by the browser. The modules scan the tree structure of the document, recognizing the conditions for their activation. If the scan was successful, the body of the module is executed, changing of the document content (tree structure). The document composition finalizes as soon as all conditions were activated.

A database of the system is a set of tools for storing and access providing to multi-format data. The body of the document loaded in the first place is stored in XML files in the file system on the server. The file system bis wrapped with a document versions tracking level to monitor the development of the document in time and create integrated backup copies. This level is quite easy to be implemented using modern version control systems like GIT or Subversion. The XML format allows to represent the logical structure of the main document body, and to store it in the document.

In order to organize standard SPARQL data source access and full-text search, storage components for the logical layer and related data as a graph of triples have been developed. ClioPatria [4] and Jena are the component implementations. The first system is interesting because it is implemented entirely with the programming languages Prolog and C. It supports several formats of compact storage of data, and tight integration of triple data with the runtime and language environment of Prolog. An [safe] access to the triples using so called Pengines protocol [5]. There are a number of implementation libraries for the protocols for popular programming languages, including client-side JavaScript, Java, Python. If for any reason an implementation of a service for storing triples by means ClioPatria is not convenient, the Java-based Jena library can be used for the service realization.

A full-text index is provided by a Elasticsearch server. The implementation of this service is simple enough because any RDF graph is representable as JSON (JSON-LD). JSON is the main format of storage of indexed information in ElasticSearch. For the representation of the resource to the user on the search result page some tripes must be marked as representative of the result. Elasticsearch has the means of fuzzy comparison of terms that allows construction of relevant information search. The main aspects of module functioning is discussed later in the applications section.

1. Analogs of the software system under development

\_ В текст добавляются аннотации при помощи специальной разметки. Семантические аннотации позволяют создавать механизмы поиска страниц Wiki, учитывающие семантику размеченного текста [6].

В отличие от Semantic MediaWiki в проекте OntoWiki аналогичные результаты получены, начиная с другой отправной точки. OntoWiki базируется на преимущественном использовании логического представления информации в виде семантической сети. Эта логическая структура редактируется при помощи генерируемых автоматически форм ввода по известным терминам в словаре. Пользователю доступно изменение лишь одного текстового свойства LOD lod:content, которое, в общем случае, содержит текст с разметкой HTML. Разметка HTML не привязывается к логической структуре отображаемого объекта (субъекта). Текст можно редактировать при помощи встроенного в OntoWiki WYSIWYG-редактора. Проект OntoWiki направлен на поддержку технологий социальных сетей на базе технологий Linked Data [6].

As was mentioned above, the used the format of the markup of the documents is a development of the project Dokieli (https://github.com/linkeddata/dokieli) [2]. The project is a WYSIWYG-editor of LOD marked up HTML pages, display style sheet (CSS) and built-in subgraph (its content is not visible on the page) presented in different formats (TTL, N3, JSON-LD, TriG). The user is able to add a new RDFa markup with editor's tools, in this case, the function of our interest is implemented in the most general form. In addition, user authentication functions are implemented according to the WebID standard with client-side JavaScript. The system extensively supports commenting (text markup) with Open Annotation (oa). Annotation engine is integrated with gitter.im service, which shows readers' comments in the form of a dialogue. All the information is stored either on servers that support the Protocol Solid (https://solid.mit.edu/), or locally in the database of a browser. The edited document be moved (copied) from one Solid-store to another one easily. The resulting texts can be downloaded to the user workstation. Thus, dokieli is a system supporting distributed text content editing, which fully implements the interaction with other LOD resources, and all these functions are implemented exclusively by means of the client-side browser JavaScript.

There is also a huge class of document editors focused on the creation of scientific publications, see, e.g., http://substance.io/. In this study, the development of tools is aimed at automating the markup of a document based on the analysis of document changes proposed in [2], and specifically in this work, at implementing one of the infrastructural problems connected with a creation of the documents on the platform web browser controlled with LOD.

1. Applications of the technologies

The tools developed for typesetting documents are used in solution of applied problems.

* 1. The texts of the curricula of the universities

Министерство образования и науки Российской Федерации после ряда экспериментов перешло на широкомасштабное внедрение Болонского процесса в образовательную среду Российской Федерации. Одной из задач, решаемых в рамках этого внедрения – переход к компетентностно-ориентированному представлению требований к педагогическому процессу. Проводимая реформа затрагивает все аспекты процесса, включая систему классификации специальностей (внедрение программ м направлений, специализация по уровням квалификации (бакалавриат, магистратура и др.), введение прикладного бакалавриата, перечень преподаваемых курсов, целей и задач курсов (согласование с компетенциями, заданными в Федеральном государственном стандарте), формы ведения занятий в вузе (введение интерактивных форм обучения), распределение лекционных и практических занятий и др. Существующая документация курса дополнена новыми формами обязательных документов – фондом оценочных средств (ФОС), аннотациями курсов. В дополнение к этому, руководство вуза с целью повышения качества предоставляемых образовательных услуг вводит собственные дополнения к форме, содержанию и требованиям к оформлению документации, в частности, к качеству конвертации в HTML для публикации на сайте вуза.

Для минимального исполнения требований руководства вузов для каждого курса преподавателю требуется оформить как минимум три документа – рабочую программу курса, аннотацию к ней, а также ФОС. Этот набор составляется для каждой возможной комбинации: вуз, кафедра, специальность, направление (профиль), программа, квалификация (бакалавр, специалист, магистрант, аспирант), академический или прикладной вариант квалификации, форма обучения (очная, заочная, вечерняя, очно-заочная и др.). Каждая комбинация отражается в учебном плане вуза, данные из которого ежегодно или два раза в год должны уточняться в рабочих программах. Последние пять лет показали уровень продуманности принимаемых менеджерами министерства решений в части требований к представлению курса – разработано четыре поколения стандартов (1,2,3 и 3+) –, для которых требовалось представить документы в соответствующей обновленной форме. На этом фоне задача разработки нового учебного пособия теперь кажется более простой, чем раньше. Опыт показывает, что преподаватели в большей части не способны справиться с качественным оформлением документов в предоставленные им временные ограничения, что приводит к необходимости руководству кафедр нанимать секретаря-специалиста, функцией которого является доведение предоставленной документации до необходимого уровня качества.

Решение данной проблемы видится в разработке программной системы, позволяющей собирать тексты рабочих программ, аннотаций и ФОС из отдельных частей: перечня компетенций и распределения нагрузки (учебный план); содержания (преподаваемых тем модуля/курса) и ФОСа, тексты которых разделяется между разными версиями документов. Титульные листы формируются также из данных учебного плана и подготовленных шаблонов. Средств Microsoft Word и Excel, стандартно используемых для решения этой задачи, со встроенным VBA явно недостаточно.

Рассмотрим схему представления размеченной рабочей программы в разработанной системе.

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html lang="ru" xmlns="http://www.w3.org/1999/xhtml"

**xmlns:taa="http://irnok.net/engine/rdfa-manipulation"**

xml:lang="ru" metal:define-macro="page">

<head> <!-- Подключение стилевых таблиц и модулей -->

</head>

<body prefix="rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns# ... foaf: <http://xmlns.com/foaf/0.1/> imei: imei.html#   
course: https://irnok.net/college/plan/01.03.02-16-1234-2461\_1%D0%BA\_PB-SM.plm.xml.xlsx-%D0%911.%D0%92.%D0%94%D0%92.3.1.html#"

resource="#post" typeof="schema:CreativeWork sioc:Post prov:Entity">

<!-- Панель управления приложением -->

<main lang="ru" resource="**#annotation**" **typeof="oa:Annotation"**

id="main-document-container">

<div **property="oa:hasTarget"   
 resource="#course-work-program"**></div>

<article **property="oa:hasBody"**   
 typeof="**schema:Article foaf:Document curr:WorkingProgram**"  
 resource="**#course-work-program**" id="main-document">

<div **taa:content="imei:title-page"**></div> <!--Титульный лист..

<div **taa:content="imei:neg-UMK"**></div> <!--Лист согласования..

<section id="contents" class="break-after">

<h2 class="nocount c">Содержание</h2>

<div id="tableOfContents"></div>

</section>

<section id="course-description" resource="#description"

**property="schema:hasPart" typeof="schema:CreativeWork"**>

<div property="schema:hasPart" resource="#purpose"  
 **typeof="dc:Text cnt:ContentAsText"** >

<div **property="cnt:chars" datatype="xsd:string"**>

<h2 **property="dc:title" datatype="xsd:string"**>Цели и задачи дисциплины (модуля)</h2>

<p>

Целю преподавания дисциплины "Технологии программирования" является освоение ...</p>

</div>

</div>

. . . . . . . .

<div property="schema:hasPart" typeof="dc:Text  
 cnt:ContentAsText" resource="#volume">

<div property="cnt:chars" datatype="xsd:string">

<h2 property="dc:title" datatype="xsd:string"> Объем дисциплины (модуля) и виды учебной работы (разделяется по формам обучения)</h2>

<div **taa:content="course:time-differ"**></div>

</div>

</div>

. . . . . . . .

В приведенном примере ключевые структуры выделены жирным шрифтом, прокомментируем эти структуры:

1. Страница, отображаемая пользователю, – это аннотация документа "#annotation", причем и аннотируемое содержимое, и текст аннотации – это одно и то же на этапе формирования документа: ресурс "#course-work-program". В LOD все ресурсы являются глобальными. В данном случае это достигается подстановкой пространства имен по умолчанию, полного URI-адреса текущей страницы, с левой стороны от названия ресурса.
2. Титульный лист и лист согласования вставляются со страницы шаблонов “imei.html” (сведения об Институте математики, экономики и информатики Иркутского государственного университета). Данные курса и название специальности вставляются в шаблон из контекста документа. Все ключевые статические шаблоны курсов можно поместить на одну страницу шаблонов.
3. Текст разбивается на разделы, обрамляется тегами <div> и <span> со соответствующими структурами RDFa. Анализ опыта разработчиков LOD ресурсов показал, что для формирования отношения достаточно использовать RDFa-теги property, typeof и datatype. От использования rel и about следует отказаться. Это делает структуру семантической разметки более строгой за счет уменьшения количества сущностей.
4. Команда taa:content добавляет в документ текст другой страницы, адрес которой формируется интерпретацией параметра, например, taa:content="course:time-differ" включает в текст страницы сгенерированную таблицу распределения часовой нагрузки между видами занятия (лекции, практики, лабораторные, СРС, и т.д.). Для генерации таких таблиц разработан сервер веб-страниц данных курсов. Каждая страница сервера представляет какой-либо курс/модуль, представленный в учебном плане. Содержание страницы кодируется структурой ее URL и сверстывается в момент первого обращения. Шаблоны текста обозначаются атрибутом id.

В режиме редактирования рабочей программы (документа) все теги, имеющие комбинацию атрибутов property="cnt:chars" и datatype="xsd:string", преобразуются в редактируемый текст. Для ясности пользователю представляется документ без taa:-включений с указанием этих включений при помощи специальных тегов и стилей. В момент завершения редактирования текст сохраняется в файловой системе сервера. Далее пользователь может выполнить фиксацию (commit) изменений и синхронизацию текста сохраненного документа с сервером системы контроля версий текста.

Исходный код проекта доступен в виде модулей языка программирования Python по следующему адресу: https://github.com/isu-enterprise. Основной модуль – isu.college.

* 1. Подготовка юридических документов

Студентами Института математики, экономики и информатики Иркутского государственного университета разработано приложение для создания юридических документов – нестандартного вида постановлений судебных исполнителей и нотариальных документов. Разметка документов в этом проекте аналогична разметке, используемой в предыдущем примере, добавлено несколько онтологий для представления юридически-значимых терминов: fibo (The Financial Industry Business Ontology) и формальных спецификаций удостоверяющих документов acrt (A Certification Ontology).

Онтологии fibo позволяют задавать роли лиц в документе, например в генеральной доверенности – кто является доверителем, а кто доверенным:

<p>Я, <span   
 **property="fibol:designatesSignatory bibo:owner"**   
 typeof="**fibol:Signatory** foaf:Person **dbr:Principal**"   
 **resource="#principal"**><span **property="foaf:name" id="signatory-name" datatype="xsd:string"** **class="edit"**>Иванова Елена Викторовна</span>,<span **property="adoc:hasPassport" resource="#signatory-passport"** **typeof="acrt:Certification"**> <span **property="acrt:qualification" resource="dbr:Passport"**>паспорт</span>…

Данный пример выражает факт, что доверитель (fibol:designatesSignatory) удостоверяется (adoc:hasPassport) при помощи паспорта (acrt:qualification dbr:Passport**)**.

В приложении добавлен режим редактирования документа в виде формы. Модуль JavaScript ищет теги содержащие набор атрибутов datatype="xsd:string" class="edit" и преобразует текст тега в поле ввода, при этом основной текст недоступен для редактирования. Основную грамматическую форму предложений текста в соответствии с форматом RDFa можно записать в атрибут content, и, например, хранить имена людей в именительном падеже, а в отображаемый текст вставлять подходящее склонение.

Согласование грамматики реализовано при помощи атрибутов id, data- и class. Значение атрибута class, равное “disp” говорит о возможности подстановки отредактированной в режиме формы строки, идентифицированной id, в текст тега. Атрибуты data-m и data-f задают вариант слов, соответствующий роду основного существительного в тексте. Атрибут data-case задает склонение (падеж и число), в которое надо поставить слова текста. Алгоритмы грамматических преобразований реализованы на сервере. Приведем пример представления текста доверенности с грамматическими конструкциями.

...проживающ<span **class="disp" id="signatory-name" data-m="ий" data-f="ая"**>\_\_\_</span> по адресу:...Подпись <span **class="disp string" id="signatory-name" data-case="gent"**>\_\_\_</span> удостоверяю...

Реализованы функции копирования документа, загрузки данных одного документа в другой. Дальнейшее совершенствование данной программной системы ведётся в направлении построения графа документооборота на основе анализа передачи данных между документами. Адрес модуля приложения - https://github.com/isu-enterprise/isu.aquarium.

1. Дальнейшее развитие программных технологий

Сегодня, большое количество информации публикуется в табличном виде в неструктурированных источниках (например, государственная статистика, финансовые и бизнес отчеты, наборы открытых научных данных, социальные медиа). Основными форматами представления данных таких источников являются PDF, растровые сканы, DJVU, а также форматы Word и Excel. Данные, хранимые в этих источниках, являются ценным ресурсом при формировании документов. Например, большинство постановлений и образовательных стандартов Минобрнауки РФ публикуется в формате PDF, причем в сканированном виде. Чтобы иметь возможность использовать данные из этих источников, прежде всего, необходимо извлечь эти данные и трансформировать их в размеченный текст или в таблицы реляционной базы данных. Для решения этой важной задачи коллективом авторов ведется разработка технологий преобразования, анализа, интерпретации, очистки и отслеживания происхождения табличных данных из неструктурированных источников, концептуализации их естественно-языкового содержания. Создана организация Cells Research Group (<https://github.com/cellsrg>), в которой реализованы два проекта: система распознавания структуры таблиц в PDF-документах (https://github.com/cellsrg/TabbyPDF); анализ семантической структуры таблиц Excel, основанный на применении правил (<https://github.com/cellsrg/cells-ssdc>). Результатом распознавания таблицы является выгрузка данных в реляционную таблицу [7].

Разметка LOD для табличных данных неструктурированных источников описывается при помощи онтологии qb (Qube ontology), которая позволяет представлять эти данные в виде гиперкуба. Для каждой ячейки таблицы в самом общем случае задается координата ячейки в кубе и свойства хранимого значения, например, тип данных и единицы измерения. Разметка таблицы в qb позволяет алгоритмически обрабатывать ее непосредственно, т.е. без необходимости предварительного анализа структуры.

Результаты предполагается внедрить в задаче публикации данных результатов исследований в области высокопроизводительного секвенирования ДНК/РНК, провидимых в Лимнологическом институте СО РАН. Для этого необходимо дополнительно размечать текстовые и табличные данные RDFa-конструкциями, поддерживаемыми системами обеспечения доступа к данным LOD в биологии, например, BIO2RDF (http://bio2rdf.org/). В результате реализации данного сервиса ЛИН СО РАН получит возможность интегрировать данные о микробиоме озера Байкал в мировую исследовательскую среду видов растений, животных и микроорганизмов аналогично [8].

Другой важной задачей анализа неструктурированной информации является автоматизация разметки однотипных документов по образцу. Задача актуальна, например, в вузах, где накоплено большое количество материала в виде учебных программ и методического обеспечения, который требует преобразования в различные формы представления: учебные программы по стандарту ФОС-3+, электронные курсы moodle и др. Кроме того, в вузах РФ активно ведутся разработки интеллектуальных систем автоматизации обучения, где материал студенту выдаётся в зависимости от предопределенного общего сценария и результатов оценки его знаний. Размеченный методический и документальный материал позволит частично автоматизировать процесс заполнения базы модулей такой системы электронного обучения.

1. Заключение

В докладе обсуждается вопрос применения технологий Linked Open Data (LOD, открытых связанных данных) для решения задач создания текстовых документов средствами, предоставляемыми современными веб-браузерами. Благодаря нотации LOD реализуется возможность интеграции данных документов и веб-приложений в один результирующий текстовый документ.

Верстка документа осуществляется при помощи алгоритмов, интерпретирующих отношения между элементами дерева HTML. Отношения описываются средствами семантического веба и LOD, в основном, конструкциями RDFa и командами, задаваемыми в атрибутах тегов HTML. Алгоритмы верстки запускаются при наличии соответствующих условий в узлах дерева документа. Каждый алгоритм вносит в дерево изменения, формирующий окончательное содержание и вид документа. Создаваемые документы хранятся на серверах, поддерживающих протокол HTTP (запросы GET и POST). Благодаря этому технология очень просто масштабируема даже на сервера хранения данных, функционирующих в режиме разделяемого хостинга и поддерживающих среду программирования PHP. Разработаны тестовые варианты хранилищ на разных языках программирования.

Приведены несколько примеров использования разработанных средств верстки для подготовки документов в образовательной среде, подготовки юридических документов и др. Дальнейшее развитие данного проекта осуществляется по нескольким направлениям: а) совершенствование средств верстки документов, б) автоматизация разметки на основе анализа изменения документов [2], в) реализация практических приложений, г) сбор дополнительной информации о потребностях пользователей, д) разработки средств обеспечения регламентированного доступа к информации. Созданные программные средства и технологии направлены на разработку программной среды глобального электронного документооборота, позволяющего физическим лицам и организациям обмениваться документами, логическая структура которых трудноформализуема в рамках традиционных подходов к автоматизации документооборота.

1. Благодарности

Результаты получены при частичной поддержке следующих проектов:

* Иркутского научного центра СО РАН № 4.1.2;
* Совета по грантам Президента Российской Федерации, государственной поддержка ведущих научных школ Российской Федерации (НШ-8081.2016.9).

Результаты получены при активном использовании сетевой инфраструктуры Телекоммуникационного центра коллективного пользования «Интегрированная информационно-вычислительная сеть Иркутского научно-образовательного комплекса» (ЦКП ИИВС ИРНОК) ([http://net.icc.ru](http://net.icc.ru/)).

Авторы благодарны сообществу Linked Opev Vocabularies (http://lov.okfn.org/dataset/lov/) за помощь, оказанную при поиске формализаций предметных областей (онтологий).

Литература

1. Bizer, Ch., Heath, T., Berners-Lee, T. Linked Data – The Story So Far. *International Journal on Semantic Web and Information Systems.* 5 (3): (2009) 1–22. doi:10.4018/jswis.2009081901. ISSN 1552-6283.
2. Черкашин Е.А., Белых П.В. и др. Подход к управлению содержанием сайта на основе технологий RDF. // *Материалы Всероссийской конференции с международным участием «Знания – Онтологии – Теории»* (ЗОНТ-2013), 8 – 10 октября 2013 г., Новосибирск. Издательство «РИЦ Прайс-курьер», 2013. 2 т., 500 с., ISSN 0568-661X
3. Capadisli, S., Guy, A., Verborgh, R., Lange, C., Auer, S., Berners-Lee, T.: Decentralised Authoring, Annotations and Notifications for a Read-Write Web with dokieli, *Procs of ICWE international conference*, 5-8 June, 2017, Rome, Italy. (to appear) [Электронный ресурс] Preprint URL:<http://csarven.ca/dokieli-rww>
4. Wielemaker, J., Beek, W., Hildebrand, M., Ossenbruggen, J. ClioPatria: A SWI-Prolog infrastructure for the Semantic Web.  [*Semantic Web*](https://www.researchgate.net/journal/1570-0844_Semantic_Web) 7(5):529-541, 2016, DOI: 10.3233/SW-150191
5. Lager, T., Wielemaker, J. Pengines: Web Logic Programming Made Easy. *Theory and Practice of Logic Programming* 14(4-5), 2014, DOI: 10.1017/S1471068414000192
6. 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. [Электронный ресурс] URL:<http://svn.aksw.org/papers/2011/COMPSAC_lod2.eu/public.pdf> (дата обращения: 30.05.2013).
7. Шигаров А.О. Восстановление логической структуры таблиц из неструктурированных текстов на основе логического вывода // Вычислительные технологии. 2014. Т. 19, № 1, С. 87-99.
8. [Dalamagas](https://arxiv.org/find/cs/1/au:+Dalamagas_T/0/1/0/all/0/1), T., [Bikakis](https://arxiv.org/find/cs/1/au:+Bikakis_N/0/1/0/all/0/1), N., [Papastefanatos](https://arxiv.org/find/cs/1/au:+Papastefanatos_G/0/1/0/all/0/1), G., [Stavrakas](https://arxiv.org/find/cs/1/au:+Stavrakas_Y/0/1/0/all/0/1), Y., [Hatzigeorgiou](https://arxiv.org/find/cs/1/au:+Hatzigeorgiou_A/0/1/0/all/0/1) A. Publishing Life Science Data as Linked Open Data: the Case Study of miRBase *Proceedings of the first International Workshop On Open Data*, WOD-2012. [Электронный ресурс]  
   URL: <https://arxiv.org/abs/1205.2320>