УДК 004.415

В. Н. Фотеева, М. Г. Пантелеев
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Построение персонализированных информационных сервисов для исследователей

Предлагается подход к построению персонализированного информационного сервиса, настраиваемого на потребности различных пользователей. Подход основан на модели информационных потребностей пользователя и модели информационного окружения. Применение подхода рассматривается на примере системы, предназначенной для определенной категории пользователей – исследователей в области семантического web. Выделены 3 базовых типа запросов: новостные, общие и аналитические. Построение сервиса рассматривается на примере основных этапов жизненного цикла запросов: их построения и выполнения. Для построения запросов используются базовая онтология исследовательской деятельности и онтология предметной области. Алгоритмы выполнения запросов включают получение данных из различных типов источников (HTML, HTML+RDFa, RSS, точки доступа SPARQL) и их обработку с учетом особенностей запроса. Представлен программный шаблон для эффективного построения модуля управления запросами, приведены направления дальнейшего развития.

Персонализированные информационные сервисы, запросы к семантическим источникам данных, информационные сервисы для исследователей

В настоящее время сохраняется тенденция высокого темпа роста объемов информации, доступной в сети Интернет, и разнообразие форм представления данных. В такой среде эффективное управление знаниями возможно благодаря персонализации информационных сервисов. Для этого сервисы должны учитывать индивидуальные запросы пользователя и уметь обрабатывать различные типы источников, в том числе семантические. Особенно важны такие сервисы для исследователей, которые тратят много времени на поиск и обработку информации, чтобы оставаться в курсе последних новостей в своей области.

Информационные потребности пользователя. При построении персонализированного информационного сервиса (ПС), который бы помогал повысить эффективность поиска и обработки информации, прежде всего необходимо выделить информационные потребности пользователей. Поскольку разные категории пользователей имеют различные информационные потребности, рассмотрим их подробнее на примере научных сотрудников.

Информационные потребности пользователя в общем случае могут быть представлены множеством информационных запросов:

$$IN = \{IQ_i\},$$

где IN – информационные потребности; IQ_i – *i*-й информационный запрос.

Далее необходимо выяснить возможные категории запросов и их особенности. На основе анализа деятельности исследователей и функционала систем-аналогов (описаны далее) выделим основные категории информационных запросов исследователей (научных работников):

1. *Новостные* – сообщать о появлении новых значимых для пользователя событий. Например: «Объявление о новой конференции», «Выход новой публикации» и т. д.

2. *Общие* – найти множество сущностей с заданными свойствами. Например: «Исследователи, работающие в области дескриптивной логики», «Проекты, связанные с технологиями семантического web за последние 3 года», «Конференции по искусственному интеллекту в 2013 г.».

3. *Аналитические* – связанные с определением статистики («Распределение (количество исследователей) в области SW по странам EU») или динамики («Рост публикаций по созданию LOD за последние 5 лет») интересующих пользователя показателей.

С учетом выделенных категорий запросов можно записать:

$$\mathrm{IN} = \left\{\mathrm{IQ}_i\right\} = \left\{\mathrm{IQ}_j^{(n)}\right\} \cup \left\{\mathrm{IQ}_k^{(g)}\right\} \cup \left\{\mathrm{IQ}_l^{(a)}\right\},$$

где $\mathrm{IQ}^{(n)}$ – новостной информационный запрос; $\mathrm{IQ}^{(g)}$ – общий информационный запрос; $\mathrm{IQ}^{(a)}$ – аналитический информационный запрос.

Аналитические запросы отличаются от новостных и общих тем, что в источниках напрямую не содержатся искомые данные, но они могут быть получены в результате выполнения операций над ними.

Жизненный цикл запроса. Для дальнейшей программной реализации обработки информационных запросов пользователей необходимо определить их жизненный цикл (ЖЦ). В общем случае в ЖЦ запроса можно выделить следующие основные *этапы*:

1) подготовка запроса;

2) выполнение (обработка) запроса;

3) представление результатов пользователю.

На этапе подготовки запроса пользователь должен сформировать запрос в соответствии с выбранной категорией и задать режимы его выполнения. Более детально подготовка запроса включает в себя:

– выбор категории запроса;

– формирование запроса;

– формирование (настройку, выбор) множества внешних источников информации;

– настройку режимов обработки запроса (частота и способ активации, методы извлечения/обработки требуемой информации и т. п.);

– задание формы представления результатов пользователю.

На втором этапе запрос обрабатывается в соответствии с заданными режимами выполнения. Выполнение (обработка) запроса в общем случае предусматривает:

– сбор информации из первичных источников;

– выделение требуемых фактов из первичных источников;

– дополнительную обработку в соответствии с типом запроса.

На последнем этапе результаты обработки запроса представляются пользователю.

На этапе формирования новостного или общего запроса пользователь должен специфицировать его атрибуты. Запрос в общем случае характеризуется двумя основными атрибутами:

1) типом нового для пользователя события (например, анонс конференции, конкурса, публикация монографии и т. д.);

2) темой события (например, семантический web, многоагентные системы).

Таким образом, для новостного запроса можно записать: $\mathrm{IQ}_{(n)j} = < \mathrm{ET}_j, \mathrm{ES}_j >$, где ET_j – тип j-го события; ES_j – тема события.

Аналогично для общего запроса: $\mathrm{IQ}_{(n)j} = < \mathrm{BT}_j, \mathrm{ES}_j >$, где BT_j – базовая сущность, выбираемая запросом.

Кроме того, на этапе формирования запроса пользователь формирует список источников, которые должны обрабатываться в каждом цикле реализации запроса.

Модель информационного окружения сервиса. Типы источников для запросов. Для того чтобы ПС мог гибко подстраиваться под различное информационное окружение, необходимо построить его модель. Эта модель описывает свойства информационного окружения, такие, как протоколы доступа к источникам, их типы, форматы получаемых данных и др. В данной статье ограничимся следующими основными типами источников для запросов:

1) новостные ленты RSS (для новостных запросов это первоочередный тип);

2) точки доступа SPARQL;

3) «чистый» HTML (под «чистым» HTML-документом понимается документ без использования микроформатов);

4) HTML + микроформаты (в частности, RDFa).

Таким образом, исходными данными новостного или общего запроса являются:

– собственно запрос $< \mathrm{ET}_j, \mathrm{ES}_j >$;

– список внешних источников.

Прототип ПС. Помогать исследователям следить за новостями в их области (новые публикации, диссертации, предстоящие конференции и др.) призваны социальные сети для научных сотрудников, такие, как [1]–[3]. Поскольку основ-

ной уклон заключается в общении и поиске исследователей со схожими интересами, такие системы имеют ограниченные возможности по настройке уведомлений о новостях и заданные форматы источников (текстовые форматы, отсутствует поддержка поиска в семантических источниках). Поиск ориентирован на внутренние базы (можно искать по набору фиксированных внешних), отсутствует возможность добавления источника для отслеживания обновлений.

Некоторые аспекты исследования и разработки прототипа ПС (в том числе обзор аналогов, общая архитектура, агентный подход к построению) были описаны в более ранних работах авторов [4]–[6]. По сравнению с другими системами главной целью при построении описываемого ПС является персонализация под конкретного пользователя, возможность расширения функционала и настройки на требуемое информационное окружение. В данной статье рассмотрены аспекты подготовки запросов (на примере новостных и общих) и сбора данных из различных типов источников, в тмо числе семантических. Как показано на рис. 1, основными модулями прототипа являются главное web-приложение, клиентский и сервисный агенты и база знаний.

Поскольку ПС призван помогать исследователям в их ежедневной работе, типы онтологий выделяются на основе анализа их деятельности.

Онтология окружения (*ОО*) описывает информационную среду, в которой работает ПС: источники данных, форматы документов и протоколы доступа к ним (формируется на основе модели ИО). Благодаря онтологии окружения агрегатора и архитектуре ПС пользователь имеет возможность выбирать необходимые ему и добавлять новые типы источников.

Базовая онтология исследовательской деятельности (*БОИД*) содержит информацию об интересующих пользователя событиях, а также описывает свойственную научно-образовательной деятельности инфраструктуру (формируется на основе категорий информационных запросов данной группы пользователей). Благодаря онтологии типов деятельности пользователь может настраивать типы отслеживаемых событий и связанных с ними данных (например, для события «публикация новой статьи» связанными являются данные об авторах).

Онтология предметной области (*ОПО*) описывает структуру конкретной предметной области и обеспечивает возможность гибкой настройки сервиса. Благодаря выделению предметной области в онтологию сервис не привязан к конкретной предметной области и может быть настроен под интересы пользователя. Для этого на этапе настройки ПС необходимо сопоставлять онтологии, что является довольно трудоемкой задачей, но может быть частично автоматизировано.

Запрос формируется с помощью описанных онтологий: из БОИД выбираются типы событий, из ОПО – темы запросов, из ОО – источники. Важным модулем является сервисный агент, поскольку именно он используется при построении запросов и сборе данных из различных типов источников.

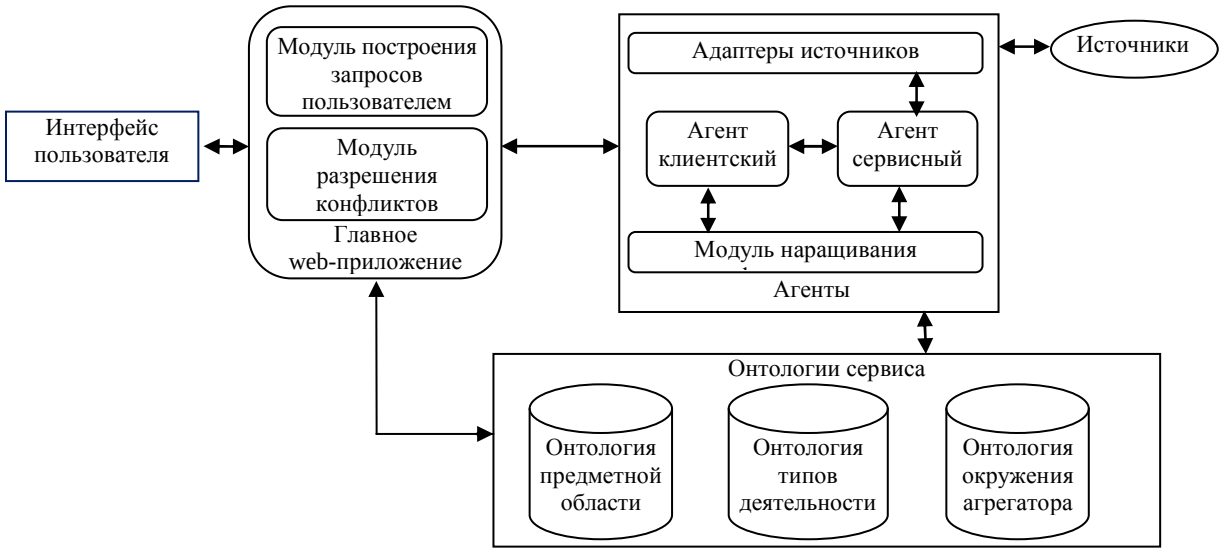

Рис. 1

Модуль управления запросами. Для эффективной обработки запросов пользователя необходимо выделить в ПС модуль, главное назначение которого:

– сделать процедуру получения ответа на запрос прозрачной вне зависимости от типа источника данных;

– обеспечить возможность подключения новых типов источников без изменения архитектуры ПС.

Назовем его модулем управления запросами (МУЗ) к источникам определенных типов. Модуль используется следующим образом:

1. При выполнении запроса. На вход модуля приходит запрос q соответствующего типа (в случае прототипа SPARQL-запрос или строка) и, если указан, тип источника или конкретный источник s. Результатом работы являются «тройки» $\{t\}$, содержащие полученные данные. Модуль определяет способ доступа к источнику конкретного типа: посылает запрос в соответствующий источник, получает ответ и приводит его к виду, подходящему для сохранения в онтологии пользователя (к «тройкам»). Модуль имеет 2 режима работы: по запросам пользователей в реальном времени и плановый сбор обновлений данных (например, раз в сутки).

2. При построении запросов пользователем к источнику определенного типа: для конфигурирования запроса и тестового извлечения данных для некоторых типов источников (в случае прототипа RSS и HTML+RDFa).

Архитектурно в прототипе этот модуль находится в сервисном агенте. Далее рассмотрим первые 2 этапа ЖЦ запросов: их построение и сбор данных из выделенных типов источников и реализацию этих этапов при помощи модуля управления запросами.

Подготовка SPARQL-запросов пользователем. Для научно-образовательных целей на основе опроса этой категории пользователей предлагается выделить типовые SPARQL-запросы с возможностью конфигурации пользователем под конкретную точку доступа. Типовые запросы пишутся в терминах онтологий ПС. Под конфигурацией понимается наложение ограничений на выбранные пользователем свойства, имеющиеся в конкретной точке доступа.

Алгоритм подготовки SPARQL-запроса:

1. Пользователь выбирает типовой запрос и точку доступа, причем словари для этой точки доступа и онтологии ПС должны быть заранее сопоставлены. В таком случае при помощи свойства «sameAs» возможно выбрать сущность, аналогичную по смыслу описанной в типовом запросе, для конкретной точки доступа.

2. Модулем управления запросами выбираются все свойства для сущности, о которой запрос, из запрашиваемой точки доступа. Для каждого запроса хранится привязка к базовому классу, экземпляры которой выбираются при его выполнении (BT для общего запроса или ET для новостного, см. ранее).

3. Пользователь выбирает свойства, на которые хочет наложить ограничения в запросе.

4. Определяются типы выбранных пользователем свойств (строка, дата, …).

5. Пользователю предлагается ввести значения выбранных свойств в качестве ограничения в запросе.

6. Готовый запрос сохраняется в БЗ ПС.

Таким образом, на выходе получается SPARQL-запрос, который может быть передан на этап сбора.

Новостные ленты (RSS). Для получения обновлений из RSS-источников используются RSS-агрегаторы. Сам по себе сбор из данного типа источника не представляет научного интереса, но так как формат очень распространен, поддержка добавлена в ПС. RSS на данный момент имеет две наиболее популярные версии: 1.0 основан на стандартах XML и RDF, в то время как 2.0 имеет более простой синтаксис и не является RDF-форматом. Для версии 2.0 возможно преобразование к RDF при помощи XSLT.

Чтобы к источникам данного типа можно было писать SPARQL-запросы, на этапе добавления источника в ПС необходимо предварительное извлечение RSS-ленты для а) сопоставления словарей (RSS-модулей) с онтологиями ПС; б) сохранения данных о содержании ленты в ПС (какие сущности и свойства содержит). Предварительное извлечение производится МУЗ. Далее алгоритм создания пользователем запроса к новостной ленте аналогичен алгоритму подготовки запросов к точкам доступа SPARQL.

HTML+RDFa. Чтобы к HTML-документам со встроенным RDFa можно было писать SPARQL-запросы, на этапе добавления источника в ПС необходимо предварительное извлечение RDF-троек из документа при помощи МУЗ. После извлечения в ПС будет сохранена информация об используемых онтологиях для описания RDFa.

Далее задача сводится к построению SPARQL-запроса. Для этого необходимо, чтобы сущности типового запроса были сопоставлены с сущностями, используемыми в RDFa. Далее алгоритм аналогичен алгоритму подготовки запросов к точкам доступа SPARQL.

«Чистые» HTML-документы. Для данного источника можно выделить 2 базовых подхода:

1) обработка с учетом структуры конкретного сайта, т. е. с учетом конкретной заранее известной HTML-разметки конкретного сайта;

2) общая обработка без учета особенностей конкретного сайта. В этом случае применимы текстовые запросы. Новостные факты должны выделяться исключительно методами text mining.

В данной статье ограничимся рассмотрением первого случая. На взгляд авторов, оптимальным вариантом будет предложить пользователю добавить на страницу разметку самостоятельно (например, RDFa или микроформаты). Для удобства необходим интерфейс, при помощи которого пользователь будет выбирать элементы HTML-страниц, содержащие конкретные данные (например, название конференции, дату начала). Такой подход сведет данный случай к обработке HTML с RDFa. Самым известным из инструментов, работающих по описанному принципу, является Structured Data Markup Helper [7] от Google (поддерживает schema.org, частично JSON-LD и микроформаты).

В ПС, чтобы задать запросы для HTML-страниц предложенным способом, выполняются следующие шаги:

1. На этапе добавления HTML-страницы пользователь размечает ее, добавляя RDFa при помощи интерфейса.

2. Разметка сохраняется для данной страницы в ПС.

3. Далее пользователь задает запросы аналогично тому, как это происходит для HTML с RDFa.

Сбор данных из SPARQL-точек доступа. МУЗ для работы с источниками данного типа должен обеспечивать выполнение SPARQL-запросов на основе заданных ET_j и Es_j с учетом особенностей конкретной точки доступа.

Алгоритм обработки источника данного типа:

1) выполнить запрос для заданной SPARQL-точки доступа;

2) получить «тройки» от SPARQL-точки доступа;

3) сохранить их в онтологии пользователя.

Для сохранения «троек» в онтологии необходимо делать выборку запросом по определенному шаблону, который обеспечивает универсальность, выбирая субъект, объект, предикат и, при наличии, текстовую метку. Приведем обобщенный SPARQL-запрос для сбора данных:

```
SELECT ?subj ?prop ?obj ?label
WHERE {
?subj a prefix:OntologyClass.  #тип события
[ ?sub jprefix:OntologyProperty ?property. #уточнение наличия свойств для наложения ограничений на значения]
?subj ?prop ?obj.
[ OPTIONAL  { ?objrdfs:label ?label.}
FILTER regex(?property, "propertyValue", "i") #уточнение значения параметров]
}
LIMIT N
```

В квадратных скобках приведены необязательные части запроса. Значения параметров уточняются различными способами в зависимости от типа значения.

Сбор данных из RSS-лент, HTML+RDFa-документов и «чистых» HTML-документов. Поскольку для конкретной RSS-ленты, HTML+RDFa документа, «чистого» HTML-документа на этапе настройки были заданы SPARQL-запросы, сбор данных из источника этого типа сводится к предыдущему пункту.

Шаблон проектирования для сбора данных. Анализ программных шаблонов показал, что для сбора данных в режиме реального времени целесообразно адаптировать шаблон «Команда» [8] (рис. 2). Для каждой операции (получение данных из источника по запросу, получение всех свойств сущности) конкретного типа источника (SPARQL-точка доступа, RSS, HTML+RDFa, чистый HTML, поисковая система, семантическая поисковая система) создаются классы-команды (например, класс для получения данных из SPARQL-точки доступа GetFromEndpointCommand). В них передаются экземпляры соответствующего класса-источника (назовем его «контроллером» этого источника, на рис. 2 для примера приведен SPARQLController). Класс SPARQLController содержит все методы получения данных из конкретного типа источника (в данном примере из SPARQL-endpoint). Класс RequestDistributor отвечает за распределение запросов по соответствующим контроллерам.

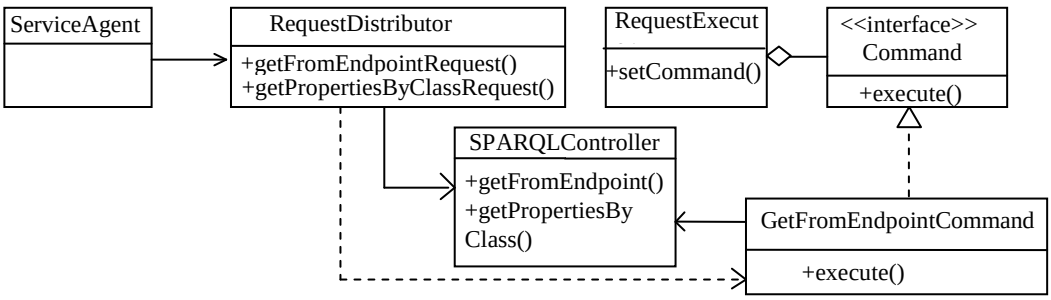

Рис. 2

Таким образом, в статье сделана попытка структуризации потребностей пользователя на примере исследователей: выделены модели информационных потребностей пользователя и информационного окружения сервиса (типов источников информации). Описанная система является прототипом нового типа персонализированных сервисов, обеспечивает сбор в различных форматах в интересах конкретного пользователя и благодаря онтологическому подходу может быть настроена на требуемое информационное окружение. Также благодаря онтологиям возможна автоматизированная обработка данных (определения «новизны» события или другой сущности для пользователя; полуавтоматическое сопоставление словарей, лежащих в основе точек доступа; разрешение конфликтов собранных данных).

В дальнейшем планируется реализация модуля разрешения конфликтов, интерфейса построения SPAQRL-запросов для пользователей, добавление возможности обработки HTML с микроданными и микроформатами. Также будет проводиться тестирование на пользователях из категокатегории научных сотрудников.

СПИСОК ЛИТЕРАТУРЫ

1. Социальная сеть для исследователей Academia.edu. URL: https://www.academia.edu/

2. Социальная сеть для исследователей Researchgate.net. URL: http://www.researchgate.net/

3. Социальная сеть для исследователей Mendeley.com. URL: http://www.mendeley.com/

4. Пантелеев М. Г., Фотеева В. Н. Построение агрегаторов научной информации для среды семантического Web. // Сб. тр. Междунар. науч.-практ. конф. «Инженерия знаний и технологий семантического веба 2012», СПбГУИТМО, 2012. С. 73–79.

5. Фотеева В. Н. Проблемы построения семантических агрегаторов // Сб. тр. 15-й Междунар. конф. по мягким вычислениям и измерениям SCM'2012. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2012. Т. 1. С. 110–112.

6. Фотеева В. Н., Пантелеев М. Г. Агрегатор семантических данных на основе агентной архитектуры // В мире научных открытий (естественные и технические науки). 2014. № 2.1. С. 505–524.

7. Structured Data Markup Helper URL: https://support.google.com/webmasters/answer/3069489

8. Head First Design Patterns / E. Freeman, E. Robson, B. Bates, K. Sierra. California: O'Reilly Media, 2004.

V. N. Foteyeva, M. G. Panteleyev
Saint-Petersburg state electrotechnical university «LETI»

CONSTRUCTION OF PERSONALIZED INFORMATION SERVICES FOR RESEARCHERS

As the amount of information available on the Internet is growing very rapidly (including linked data, in particular LOD), the creation of customized tools for knowledge management is becoming increasingly important. The paper proposes an approach to the construction of personalized information services that can be customized to the needs of different users. The approach is based on two models: model of user's information needs and information environment model (types of data sources). An implementation of the approach is considered on the example of system which is intended for concrete category of users – researchers in the area of the Semantic Web. Three basic types of user's queries have been identified: news, general and analytical. Designing the personalized service is considered from the standpoint of the main stages of the queries lifecycle: its construction and execution. Two types of ontologies are used for initial query constructing: basic ontology of research activity and the do-main ontology. Query execution algorithms include obtaining data from different types of sources (HTML, HTML+RDFa, RSS, SPARQL-endpoint) and its processing depending on the features of the query. In addition the design pattern for effective building of queries management module is proposed. In conclusion future directions of prototype improvement are discussed.

Personalized information services, queries to semantic data sources, information services for researchers