Использование технологии Semantic Web в системе поиска несоответствий в текстах документов

Андреев А.М. Березкин Д.В. Рымарь В.С. Симаков К.В.

НПЦ «ИНТЕЛТЕК ПЛЮС»
arka@inteltec.ru

Аннотация

В статье изложен ряд технологических решений, принятых при разработке системы выявления несоответствий в текстах редактируемых документов. Для выявления несоответствий в качестве эталона система использует онтологию предметной области. Приведена функциональная схема системы, описана логика работы основных модулей.

Введение

Быстро увеличивающееся количество электронных документов на естественных языках поставило задачу выявления несоответствия информации, содержащейся в них, некоторому эталонному описанию предметной области.

В отличие от предыдущей работы [16], где основной упор делался на выявление противоречий в юридических документах, в данной работе решается задача выявления в текстах документов несоответствий, таких как: ошибочные должности сотрудников организаций, ссылки на устаревшие структурные подразделения организаций, неправильные телефонные номера должностных лиц. Выявление несоответствий являются частным случаем поиска противоречий в текстах документов, при этом модель несоответствия является упрощенным вариантом модели противоречия. Несоответствие в данном случае трактуется как неэквивалентность факта, выявленного при анализе текста, имеющимся в базе знаний фактам.

Обзор существующих технологий и стандартов

Задачи хранения, извлечения, получения и анализа знаний решаются в рамках направления, получившего названия Semantic Web [2]. В обоих случаях необходимо иметь структурированные хранилища информации и множества правил вывода, которые компьютеры бы использовали для автоматических рассуждений [3]. Учитывая многолетний опыт разработки, стандартизации и развития технологий Semantic Web в рамках World Wide Web Consortium[4], использование существующих разработок в этой области является закономерным.

Сейчас уже создан ряд важнейших технологий (представленных на рисунке 1): Расширенный Язык Разметки (Extensible Markup Language, XML) [5], Система Описания Ресурсов (Resource Description Framework, RDF)[6], Язык Сетевых Онтологий (Ontology Web Language, OWL)[1], используемых для описания, хранения и распространения знаний. Так же стоит выделить SPARQL Язык запросов к RDF (SPARQL Query Language for RDF)[7], который 6 апреля 2006 года стал кандидатом к рекомендации W3C, так же ведутся работы по стандарту протокола SPARQL для RDF и стандарта, определяющего XML-формат представления результатов обработки SPARQL-запросов.

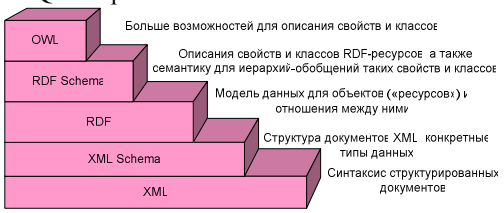

Рис. 1. Рекомендации W3C касательно Semantic Web.

Применение существующих стандартов и технологий позволит использовать уже готовые программные продукты и компоненты сторонних разработчиков. При этом в дальнейшем с развитием технологий Semantic Web можно ожидать появления еще большего числа разработок в этой области.

Система семантического контроля текстов редактируемых документов

Одна из основных функций такой системы – выделение в текстовом документе (например, в MSWord) словосочетаний с предполагаемым нарушением семантических связей. Эту же задачу можно сформулировать и как поиск противоречий между знаниями, выявленными в тексте и знаниями, хранящимися в онтологии.

Предложена система, решающая указанные задачи, структура которой представлена на рисунке 2. На рисунке 2 показаны АРМы пользователей системы, а также отмечены средства разработки отдельных модулей системы.

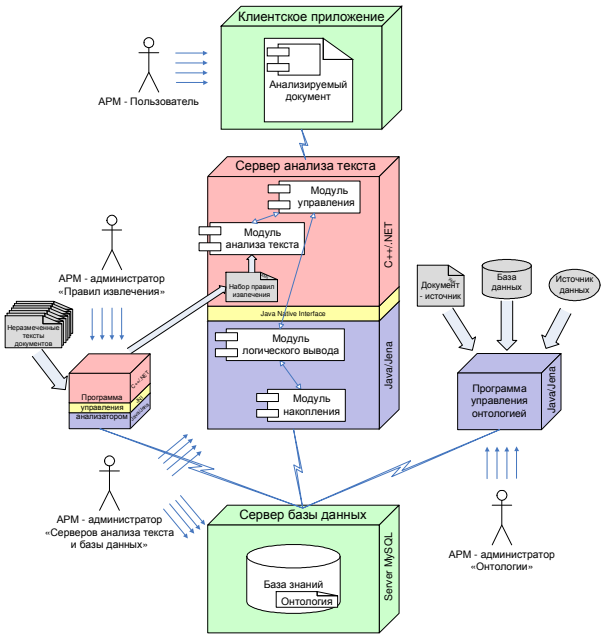

Рис. 2. Структура системы семантического контроля текстов.

С точки зрения работы с системой можно выделить 4 АРМ:

- Пользователь,
- Администратор «Правил извлечения»,
- Администратор «Онтологии»,
- Администратор «Серверов анализа текста и базы данных».

АРМ – администратор «Серверов анализа текста и базы данных». Функции этого АРМ сводятся к первоначальной настройке (или дополнительной настройке) серверов базы данных и анализа текста.

АРМ – администратор правил извлечения. Задача данного АРМ – формирование правил извлечения, используемых модулем анализа текстов сервера.

АРМ – администратор «Онтологии». Основная функция – управления онтологией. В частности создание, удаление, наполнение, выгрузка области онтологии. Функции:

АРМ – Пользователь. Пользователи системы работают в текстовом редакторе (например, MS Word). С помощью этого АРМа конечные пользователи реализуют такие функции системы как:

автоматическое (по команде пользователя) выделение в тексте терминов и словосочетаний, присутствующих в онтологии;

выделение в тексте словосочетаний с предполагаемым нарушением семантических связей;

просмотр в онтологии терминов и словосочетаний, выделенных в тексте автоматически или пользователем;

В текстовом редакторе, пользователю необходимо нажать соответствующую какой - либо функции кнопку. Далее система в автоматическом режиме выполнит запрос к серверу анализа текста и по результатам его выполнения внесет соответствующие выделения в текст или выведет справочную информацию.

Центральной частью системы является «Сервер анализа текста». Обращение к этому серверу происходит из программы работы с текстом, которая подает на обработку серверу рабочий текст пользователя. После выполнения анализа сервер возвращает тот же текст с указанием форматирования отдельных слов, словосочетаний и фраз. Пользователю, работающему с документом в текстовом редакторе, подсвечиваются словосочетания с предполагаемым нарушением семантических связей. Среди дополнительных функций можно выделить выдачу справок на основе информации, имеющейся в базе знаний. Рассмотрим подробнее функции и способы реализации каждой из частей.

Функция программы работы с текстом сводятся к передаче текста документа серверу анализа текста, получению от него результата анализа и форматирование фрагментов текста, на которых следует заострить внимание (выделение цветом, подчеркивание и т.п.). Дополнительно эта программа посылает запросы к серверу анализа текста на выдачу справок на основе информации, имеющейся в онтологии.

Сервер анализа текста имеет одну точку подключения для взаимодействия с программой работы с текстом и прикладными программами администрирования и управления. Модуль управления сервера анализа текста имеет ряд функций. Во-первых, это взаимодействие с программой работы с текстами и прикладными программами управления и администрирования. Во-вторых, это ведение журнала транзакций. Заметим, что проблемы безопасности и доступа к информации решаются на уровне сервера базы данных на основе учетных записей пользователя. На этот модуль возложены функции по обеспечению взаимодействия остальных модулей сервера и распределения нагрузки.

Текст, поступая в модуль управления, передаётся в модуль анализа текста. В настоящей работе мы не будем рассматривать вопросы функционирования этого модуля, а также метод извлечения фактов из текстов, который в нем реализован. Этим вопросам посвящен отдельный представленный на конференцию доклад авторов «Модель извлечения фактов из естественно-языковых текстов и метод ее обучения», здесь же мы сосредоточимся на вопросах, связанных с использованием онтологий для анализа текстов. Отметим лишь, что в этом модуле анализа текста происходит выявление знаний в тексте на основе pattern-based модели, основанной на образцах.

Модель знаний системы основана на онтологии, записанной на языке OWL. Базовый строительный блок модели данных – утверждение, представляющее собой тройку: ресурс (экземпляр класса для OWL), именованное свойство и его значение. В терминологии RDF эти три части утверждения называются соответственно: субъект, предикат и объект [6]. Под свойством следует понимать некий аспект, характеристику, атрибут или отношение, ис-

пользуемое для описания ресурса. Каждое свойство имеет свой специфический смысл, допустимые значения, тип ресурсов, к которым оно может быть применено, а также отношения с другими свойствами. Согласно спецификации, значение свойства может иметь один из двух типов. Первый – это ресурс (экземпляр некоторого класса), задаваемый некоторым URI. Второй тип – литерал – есть некоторое текстовое значение характеристики. Впрочем, литерал может выражать собой значение любого примитивного типа данных, присутствующего в XML. Таким образом, свойства P языка OWL являются бинарными отношениями на C×V или C×C, где C являются классами, V литеральными значениями. Свойства первого типа связывают экземпляры классов с конкретными значениями, свойства второго типа связывают экземпляры классов друг с другом. Задача извлечения знаний при таких условиях сводится к созданию экземпляров классов, распознаванию значений и распознаванию свойств в естественно-языковых текстах. В общем случае в текстах можно найти упоминание о том или ином экземпляре класса только в рамках некоторого свойства. В этом случае удобно говорить о роли экземпляра в рамках свойства.

С учетом OWL ограничения о бинарности свойств, экземпляр класса может играть либо роль субъекта свойства, либо роль объекта. Для свойств типа C×V экземпляр может выступать только в роли субъекта. Таким образом, извлекаемыми элементами являются либо тройки вида <i,p,t>, либо тройки вида <c1_i,p,c2_j>. Тройка <c_i,p,t> указывает на наличие в тексте свойства p типа C×V некоторого экземпляра i класса c, выраженного текстовым фрагментом t. Тройка <c1_i,p,c2_j> указывает на наличие свойства p типа C×C между извлеченным ранее экземпляром i класса c1 и экземпляром j класса c2.

Извлеченные тройки передаются модулю управления для последующей передачи модулю логического вывода. На рисунке 3 показан пример небольшого фрагмента текста подвергнутого анализу.

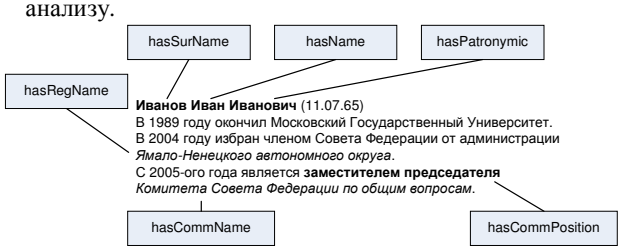

Рис. 3. Пример анализа текста.

Необходимо сразу отметить, что приведенные примеры являются чисто иллюстративными и их содержание служит лишь для пояснения рассматриваемых авторами технологий.

Результат работы модуля анализа текста над приведенным на рис. 3 примере представлен на рисунке 4.

```
<Functioner_1,hasName,"Иван">
<Functioner_1,hasSurName,"Иванов">
<Functioner_1,hasPatronymic,"Иванович">
<CommitteeMember_1,hasCommName,"Комитет по общим вопросам">
<CommitteeMember_1,hasCommPosition,"заместитель председателя">
<Region_1,hasRegName,"Ямало-Ненецкий автономный округ">
<isRepresentativeOf,Functioner_1,Region_1>
<isMemberOf,Functioner_1,CommitteeMember_1>
```

Рис. 4. Факты, выявленные из текста.

Особенности проектирования онтологии системы

Необходимо отметить, что для заполнения базы знаний в системе используются несколько источников информации. Каждый из этих источников отвечает за отдельную локальную область разрабатываемой онтологии. Эти источники разнородны по форматам представления данных: различные базы данных, XML файлы и т.д., а также по ее семантике. Возможен режим работы, когда какие-то области формируются оператором. Ответственными за источники информации являются различные структурные подразделения организации или даже разные организации, что и определяет их большую разнородность. Логическое разделение онтологии на области проиллюстрированы на рисунке 5.

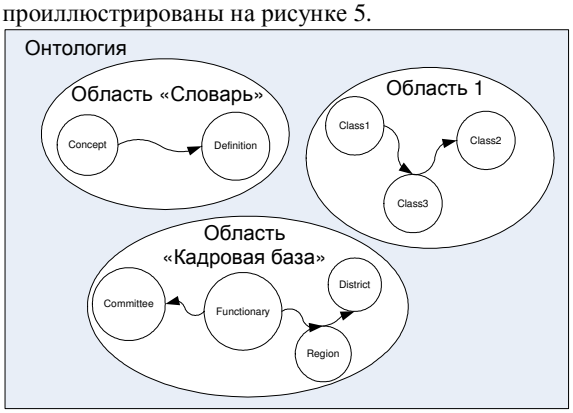

Рис. 5. Онтология логически разделена на локальные области.

В рамках данной статьи рассматривается локальная область «Кадровая база». Для нее в качестве источника используется XML файл. На основе анализа этого источника данных были выявлены основные понятия данной предметной области и связи между ними, далее определены для них классы и создана таксономия классов. На рисунке 6 представлен фрагмент таксономия классов предметной области «Кадровая база».

Заметим, что каждый экземпляр в мире OWL является членом класса owl:Thing. Таким образом, каждый определенный нами класс автоматически является подклассом owl:Thing. На рисунке 7 представлен фрагмент онтологии «Кадровая база» без учета таксономии классов.

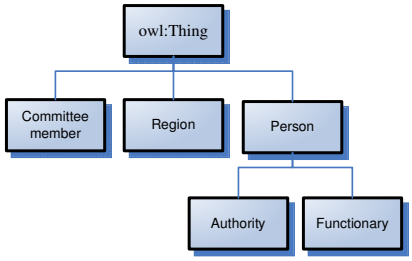

Рис. 6. Фрагмент таксономии классов.

Схема онтологии создается для каждой предметной области в ручном режиме с помощью редактора онтологии, который будет описан ниже. Созданные таким образом онтологии используются программами загрузчиками онтологий, которые добавляют отдельные экземпляры классов и их свойства.

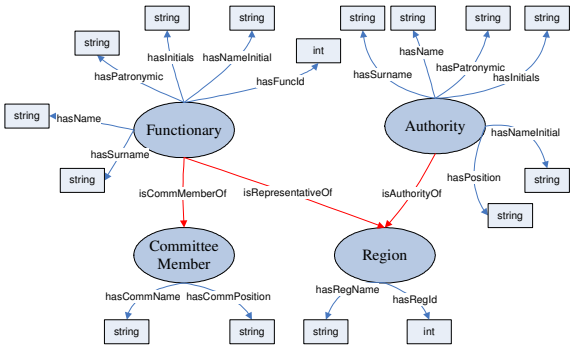

Рис. 7. Пример онтологии «Кадровая база».

Программа загрузчик использует определенный источник информации для наполнения онтологии, при этом каждая программа «ведет» свою поименованную область онтологии.

Модуль логического вывода

Основной задачей системы является выявление несоответствий извлеченных фактов и фактов, содержащихся в базе знаний. Она реализуется модулем логического вывода, который производит сопоставление фактов, извлекаемых из текста, фактам, содержащимся в базе знаний.

Рассмотрим подробнее логику работы модуля логического вывода. Как уже отмечалось, на вход поступает несколько вариантов извлечения (набор цепочек фактов). Необходимо рассмотреть все варианты извлечения и на основе наиболее «подходящего» сделать вывод о достоверности фактов. Как уже отмечалось, из текста извлекаются цепочки троек двух типов – факты, описывающие свойства – значения и свойства – объекты. Тогда наш анализ можно разделить на несколько стадий:

Опираясь на свойства – значения идентифицируем в онтологии экземпляры. Факты, описывающие такие свойства, имеют вид <Class_Id,PropertyId,Value>. Тогда в онтологии необходимо найти экземпляр класса с именем Class, у которого свойство – значение с именем PropertyId имеет значение Value. Заметим, что для некоторых экземпляров может существовать несколько определяющих их свойств – значений. Например,

<Functionary_1,hasName,”Иванов”>

<Functionary_1,hasPatronymic,” Иванович”>

<Functionary_1,hasSurName,”Иванов”>

В таких ситуациях выполняется поиск экземпляра, у которого описанные свойства имеют соответствующие значения. С другой стороны для некоторого количества свойств-значений может быть идентифицировано несколько экземпляров какого-то класса. Например, по тройке <Functionary_1,hasSurName,”Иванов”> может быть найдено несколько экземпляров класса Functionary (функционер), у которых свойство hasSurName (имеет фамилию) имеет значение ”Иванов”.

Для всех идентифицированных экземпляров проверяем справедливость свойств – объектов, имеющих вид <Class1_Id1,PropertyId,Class2_Id2>. Каждый такой факт устанавливает связь (свойство – объект с именем PropertyId) между двумя экземплярами разных классов. Если на первом шаге было выявлено несколько экземпляров, необходимо произвести такой поиск для всех возможных вариантов.

Заметим, что на первом шаге можем идентифицировать не все объекты, тогда на втором шаге проверяем справедливость свойств – объектов только для найденных экземпляров. Рассмотрим подробнее анализ одной цепочки троек. На рисунке 8 проиллюстрирован алгоритм работы модуля логического вывода и показаны возможные варианты при рассмотрении одной цепочки фактов (варианта извлечения).

Рассмотрим каждый из вариантов:

1 – Найдены все экземпляры, но не все связи между ними подтверждены. На основании только этого варианта извлечения можно сделать вывод о том, что в анализируемом фрагменте текста существует возможное нарушение семантических связей, дальнейший анализ этого фрагмента можно не проводить.

2 – Найдены все экземпляры, и все связи между ними подтверждены. На основании только этого варианта извлечения можно сделать вывод о том, что в анализируемом фрагменте текста нет нарушения семантических связей, т.е. все извлеченные факты подтверждены фактами из онтологии. И дальнейший анализ такого фрагмента можно не проводить.

3 – Не найдено ни одного экземпляра. Это говорит либо о неправильном извлечении фактов из текста, либо о неактуальности онтологии (онтология неполная или содержащаяся в ней информация устарела). В любом случае это необходимо зафиксировать в Log-файле для дальнейшего анализа.

4 – Найдены не все экземпляры, и при этом не все связи между ними подтверждены. То, что не найдены все экземпляры говорит о неактуальности онтологии или об ошибках извлечения. Это необходимо зафиксировать в Log-файле для

дальнейшего рассмотрения, но при этом дальнейший анализ можно проводить по найденным экземплярам. Т.к. не все связи между найденными экземплярами установлены, то предположительно в анализируемом фрагменте текста существует нарушение семантических связей. Т.к. идентифицированы не все экземпляры, необходим дальнейший анализ этого фрагмента, т.е. необходимо рассмотреть другие варианты извлечения.

5 – Найдены не все экземпляры, но все связи между ними подтверждены. То, что не найдены все экземпляры говорит о неактуальности онтологии или об ошибках извлечения. Это необходимо зафиксировать в Log-файле для дальнейшего рассмотрения, но при этом дальнейший анализ можно проводить по найденным экземплярам. Т.к. все связи между найденными экземплярами установлены, то предположительно в анализируемом фрагменте текста нет нарушения семантических связей. Т.к. идентифицированы не все экземпляры, необходим дальнейший анализ этого фрагмента, т.е. необходимо рассмотреть другие варианты извлечения.

Теперь рассмотрим, как сделать вывод о нарушении или отсутствия нарушения семантических связей для всего набора цепочек.

Если при анализе одного варианта извлечения (одной цепочки) получается вариант 1 или 2, то можем сделать вывод обо всем рассматриваемом фрагменте текста и другие цепочки не рассматривать. Другой вариант, если при анализе всех цепочек получали только варианты 3, 4 или 5. Получение варианта 3 для всех цепочек говорит об ошибках извлечения или неактуальности онтологии, делать вывод на основе такого анализа невозможно, поэтому просто фиксируем этот факт в Log-файл. Если при рассмотрении всех цепочек (всех вариантов извлечения) для данного фрагмента встретился хотя бы один вариант 5 (т.е. остальные 3 или 4), то делаем вывод, что предположительно в данном фрагменте нет нарушения семантических связей. При наличии только вариантов 4 и, возможно, некоторых 3, делаем вывод о наличии возможного нарушения семантических связей. Таким образом, рассмотрены все возможные варианты, полученные при анализе каждой из цепочек.

Остановимся подробнее на формировании SPARQL запросов для поиска в онтологии модулем накопления. Рассмотрим два вида запросов: для свойств – значений и свойств – объектов.

Запрос для тройки, описывающей свойство – значение. Например, для тройки, извлеченной из фрагмента текста <Functionary_1,hasName,”Иванов”> сформируем SPARQL запрос к онтологии. Нам необходимо найти экземпляр или экземпляры класса Functionary, у которых свойство hasName имеет значение «Иван».

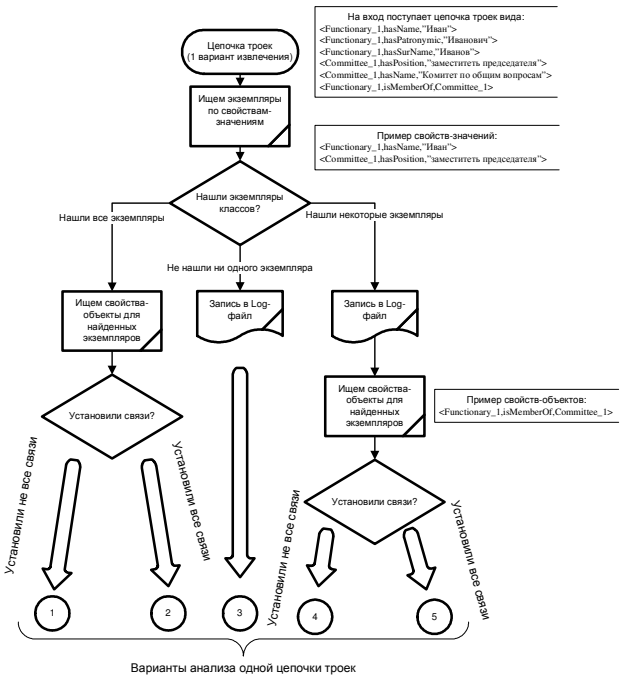

Рис. 8. Работа модуля логического вывода.

Это можно выполнить, например, с помощью следующего запроса:

```
SELECT ?x  WHERE
{
  ?x rdf:type <Functionary> .
  ?x < hasName> \"Иван\"^^xsd:string.
}
```

В ответ на этот запрос мы получим список имен экземпляров класса Functionary, например, Functionary_765, Functionary_8765, Functionary_8766. Или ничего, если нет ни одного экземпляра, у которого свойство hasName имеет значение ”Иван”.

Запрос для тройки, описывающей свойство – объект. Например, для тройки <Functionary_765,isMemberOf,Committee_2424> сформируем SPARQL запрос к онтологии. Это можно выполнить, например, с помощью запроса:

```
ASK WHERE
{
  Functionary_765
    <isMemberOf>
  Committee_2424.
}
```

В ответ на этот запрос получим ответ в формате «YES»/«NO», что говорит о наличии/отсутствии такой связи (свойства-объекта) в онтологии.

Таким образом, цепочку троек свойств – значений делим на несколько частей, каждая из которых описывает один конкретный экземпляр. Затем, на основе каждой из частей генерируем SPARQL [7] запрос, который исполняется с помощью модуля накопления. В ответ на каждый запрос получаем список экземпляров (возможно пустой). После этого для всех свойств – объектов формируем запросы со всеми возможными вариантами найденных экземпляров. После выполнения всех таких запросов можем присвоить этой цепочке троек один из

номеров варианта решения (от 1 до 5, см. рисунок 8). Исходя из варианта решения для текущей цепочки, анализируем остальные цепочки троек или делаем вывод о возможном нарушении семантических связей.

Таким образом, на основе набора вариантов извлечения модуль логического вывода принимает решение о возможном нарушении семантических связей.

Другие модули системы

Сервер базы данных выполняет две основные функции. Во-первых, хранение/обновление онтологии и выполнение к ней запросов. Во-вторых, разграничение прав доступа к онтологии. Так же важными функциями является ведение журнала доступа к серверу базы данных и возможность резервного копирования базы знаний.

На сервере анализа текста предусмотрен так же модуль накопления. Он решает две задачи: обеспечивает взаимодействие с сервером базы данных, а так же формирует запросы для работы с онтологией. Это позволит использовать базы сторонних разработчиков с минимальными изменениями в коде сервера анализа текста, локализованными в одном модуле.

Так же существуют независимые прикладные программы. Например, программы управления онтологиями, предназначенные для ведения онтологии: создание, изменение, удаление. Каждая программа ведёт свою область онтологии, опираясь на конкретный источник. В качестве источников информации в общем случае могут выступать структурированные текстовые файлы (XML, HTML), базы данных, редакторы онтологий и т.д.

Таким образом, мы описали полный цикл работы системы – от загрузки и управления онтологией, до непосредственного выявления неэквивалентности фактов в тексте и в базе знаний.

Выбор средств разработки

Для эффективной разработки готовых систем в современных условиях с жесткими требованиями по времени необходимо использовать готовые компоненты и модули сторонних разработчиков. Особенно это актуально при использовании стандартизированных технологий. В нашем случае это работа с онтологией на языке OWL, выполнения запросов на языке SPARQL к RDF нотации этой онтологии. Для возможности использования компонентов сторонних разработчиков, необходимо чтобы они удовлетворяли нескольким требованиям: обладать API для работы с OWL онтологией, иметь возможность выполнять SPARQL запросы, иметь парсер для работы с онтологией, а так же использовать для хранения и работы с онтологией СУБД. Всем этим требованиям удовлетворяет Jena [9], разработанная HP Labs[10]. Jena – это набор свободно распространяемых библиотек, написанных на Java.

Jena включает:

- RDF API;
- Чтение/запись документов RDF в RDF/XML формате;
- OWL API;
- Хранилища онтологий в памяти и в СУБД;
- Выполнение SPARQL запросов.

В качестве СУБД, при работе с Jena, предлагается использовать MySQL, Oracle, PostgreSQL. В данной разработке сделан выбор в пользу MySQL [11], как надежной, быстродействующей, свободно распространяемой СУБД.

Таким образом, в качестве платформы разработки выбирана Java, в качестве сервера базы данных используется MySQL. Доступ к этой СУБД из Java программ осуществляется стандартными средствами MySQL (JDBC). Для работы с онтологиями используются средства Jena, библиотеки которой позволяют выполнять все необходимые нам операции. Заметим, что библиотеки Jena используются в модуле накопления сервера анализа текста и в аналогичных модулях прикладных программ управления онтологией.

В системе также имеется возможность создания и ведения области онтологии в ручном режиме с помощью редакторов онтологии. Существуют как коммерческие, так и свободно распространяющиеся редакторы онтологий. Среди платных стоит выделить Altova's SemanticWorks[12], среди бесплатных - Protégé [13].

Как уже отмечалось, с развитием технологий, относящихся к Semantic Web, следует ожидать появления новых и дальнейшего развития уже существующих средств разработки.

Заключение

В настоящее время создана и внедряется в Совете Федерации Федерального Собрания Российской Федерации первая очередь информационной системы «Семантический контроль текстов редактируемых документов», описанная в данной статье. Она используется специалистами Управления информационного и документационного обеспечения аппарата Совета Федерации для проверки правильности расшифровки стенограмм и проверки редакций различных типов документов на предмет их соответствия эталонным словарям и базам данных. Онтология, используемая системой, создана на основе кадровой базы Совета Федерации, в настоящий момент содержит 6 классов, общее количество экземпляров по всем 6-ти классам составляет 1183, суммарное количество свойств-значений по этим экземплярам – 3785, общее количество свойств-объектов, связывающих экземпляры онтологии, – 755.

В качестве направления развития системы можно указать на необходимость использовать более сложные механизмы логические вывода и анализа, например, языков логического вывода (SWRL[14], RuleML[15]). Также увеличение сложности и

объемов хранимых в онтологии знаний, добавление новых логических областей, очевидно, потребует совершенствования средств работы с онтологией.

Литература

[1] Web Ontology Language (OWL), http://www.w3.org/2004/OWL/
[2] Semantic Web, http://www.w3.org/2001/sw/ .
[3] Tim Berners-Lee, James Hendler and Ora Lassila, The Semantic Web, http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&catID=2.
[4] The Extensible Markup Language (XML), http://www.w3.org/XML/
[5] Resource Description Framework (RDF), http://www.w3.org/RDF/
[6] SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/
[7] Владимир Шрайбман, Выражение семантики данных. RDF против XML, http://www.citforum.ru/internet/xml/rdf_xml/
[8] World Wide Web Consortium (W3C), http://www.w3.org/
[9] Jena – A Semantic Web Framework for Java, http://jena.sourceforge.net/
[10] HP Labs Semantic Web Research, http://www.hpl.hp.com/semweb/index.html
[11] MySQL, http://www.mysql.com
[12] SemanticWorks, http://www.altova.com/products_semanticworks.html
[13] Protégé, http://protege.stanford.edu/
[14] SWRL: A Semantic Web Rule Language Combining OWL and RuleML, http://www.w3.org/Submission/SWRL/
[15] The Rule Markup Initiative, http://www.ruleml.org/
[16] Андреев А.М., Березкин Д.В., Симаков К. В. Особенности проектирования модели и онтологии предметной области для поиска противоречий в правовых электронных библиотеках. 6-ая Всероссийская научная конференция RCDL'2004.

Using Semantic Web technology for the task of inconsistency detection in natural language texts

The primary purpose of this article is to show main technological decisions was made during developing the system of inconsistency detection in natural language texts of some domain. For this task we use the ontology as etalon knowledge base. We present the architecture of the system and describe the logic of its working.