Московский Государственный Университет им. М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра системного программирования

Дипломная работа

Исследование и разработка методов извлечения отношений из текстов на основе онтологии, структурирующей данные проекта Википедия

Выполнила:
Астапова Оксана Петровна
Студент 527 группы

Научный руководитель:
к.ф.-м.н. Турдаков Денис Юрьевич

Москва, 2014

Аннотация

Необходимость структурировать информацию и извлекать из нее факты является одной из важных задач для современного общества. Одним из наиболее часто используемых способов структурирования информации является представление ее в виде триплетов «объект-предикат-субъект», где предикат описывает отношение между объектом и субъектом. Сложность определения отношений заключается еще и в том, что на естественном языке для описания одного и того же по смыслу отношения могут быть использованы совершенно различные конструкции.

В дипломной работе рассматриваются современные способы извлечения отношений из текстов на естественных языках.

Разработан метод извлечения отношений, описывающих заданные предикаты, из текстов на русском языке с использованием DBpedia – онтологии, структурирующей данные проекта Википедия. Для его реализации построена система, представляющая собой совокупность модулей обработки текста и баз данных. Разработанный метод основан на построении шаблонов и может быть использован для обогащения базы знаний, поиска фактов в текстах, улучшения структуры баз знаний (в частности, DBpedia).

Содержание

Введение

Особенности естественных языков таковы, что одну и ту же информацию можно представить многими способами даже в рамках одного языка. Естественным образом возникает необходимость некоторого унифицированного представления и структурирования текста. Как только информация, представленная в тексте, становится упорядоченной и привязанной к некоторой системе идентификаторов, появляется возможность для автоматической работы с ней: извлечения фактов, поиска нужной информации, поиска похожих текстов и прочих задач.

Эти идеи привели к возникновению **семантической паутины** (Semantic web) - направления развития Всемирной паутины, целью которого является представление информации в виде, пригодном для машинной обработки. Термин «семантическая паутина» был впервые предложен сэром Тимом Бернерсом-Ли (изобретателем Всемирной паутины) в 2001 году [1] и называется им «следующим шагом в развитии Всемирной паутины».

Семантическая паутина — это надстройка над существующей Всемирной паутиной, задачей которой является делать размещённую в ней информацию пригодной для машинной обработки, благодаря двум основным ее характеристикам:

1. *Использованию унифицированных идентификаторов ресурсов (**URI**)*, называемых также адресами. URI глобально уникальны и используются не только для ссылок на объекты, но и для их именования. Свои URI в семантической паутине есть не только у веб-страниц, но и у объектов реального мира (людей, городов, художественных произведений и так далее), а также у абстрактных понятий (например, у свойств «имя», «должность», «цвет»). Пример URI, соответствующего понятию «оранжевый цвет» в проекте DBpedia: http://DBpedia.org/page/Orange_(colour)

2. *Использованию семантических сетей и онтологий.* **Семантическая сеть** – это информационная модель предметной области, имеющая вид ориентированного графа, вершины которого соответствуют объектам предметной области, а дуги (рёбра) задают отношения между ними. Объектами могут быть понятия, события, свойства, процессы [2]. Таким образом, семантическая сеть является одним из способов представления знаний. В семантической паутине узлы и дуги семантической сети имеют URI и описываются при помощи стандарта RDF.

Онтология представляет собой формализацию некоторой области знаний при помощи концептуальной схемы (информация представляется как иерархическая структура понятий). Как правило, такая схема включает в себя описание понятий, их свойств, отношений между ними и некоторой логики (набора правил, теорем, ограничений), ограничивающей возможные свойства и отношения для различных понятий.

Язык онтологии – это формальный язык, описывающий онтологию. Для описания онтологий, используемых для работы с семантическими сетями, используется рекомендованный консорциумом W3C язык OWL[1], который можно считать расширением языка RDF.

Resource Description Framework (RDF, «среда описания ресурса») — это модель для представления данных и метаданных, разработанная консорциумом W3C. RDF представляет утверждения о ресурсах в виде, пригодном для машинной обработки. Актуальная спецификация модели представлена в на странице консорциума[2]. Ресурсом может быть любая сущность, как информационная (страница, изображение), так и неинформационная (например, человек, город, некое абстрактное понятие).

Утверждение, высказываемое о ресурсе, имеет вид «*субъект — предикат — объект*» и называется **триплетом**. Утверждение «небо голубого цвета» в RDF-терминологии можно представить следующим образом: субъект — «небо», предикат — «имеет цвет», объект — «голубой». Для обозначения субъектов, предикатов и объектов в RDF используются URI.

Рисунок 1: Триплет RDF

Множество RDF-утверждений образует ориентированный граф, в котором вершинами являются субъекты и объекты, а рёбра помечены предикатами.

RDF предоставляет средства для построения информационных моделей, но не касается семантики описываемого. Взятый в отдельности граф RDF можно понимать только как граф. Толкование значения зависит от интерпретации URI и структуры графа пользователями.

[1] http://www.w3.org/TR/2004/REC-owl-semantics-20040210/
[2] http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140225/

Как уже было сказано выше, RDF является частью концепции семантической паутины.

Несмотря на то, что идея создания некой глобальной базы знаний считается нереализуемой полностью и критикуется за неизбежное дублирование информации при описании метаданных – семантическая паутина была признана консорциумом W3C и послужила базой для развития некоторых важных проектов по структурированию информации на естественных языках.

Одним из таких проектов стала DBpedia [4].

DBpedia — проект, направленный на извлечение структурированной информации из данных, созданных в рамках проекта Википедия. DBpedia позволяет пользователям запрашивать информацию, основанную на отношениях и свойствах ресурсов Википедии, в том числе ссылки на соответствующие базы данных. Проект был начат группой разработчиков из Свободного университета Берлина и Лейпцигского университета, в сотрудничестве с OpenLink Software, и впервые был опубликован в 2007 году. Данные проекта доступны на условиях свободной лицензии. На момент написания данной работы в DBpedia представлены данные на 119 языках, всего около 24.9 млн сущностей. Онтология содержит 2.46 * 10^9 RDF-триплетов, из которых 470 млн были получены из английской версии Википедии, 1.98 млрд – из статей на других языках, и около 45 млн являются ссылками на внешние наборы данных. DBpedia была выбрана для использования в дипломной работе, поскольку является полностью открытым проектом, постоянно обогащается новыми данными из Википедии, хранит информацию на русском языке и охватывает достаточно широкий спектр предметных областей.

В дипломной работе рассматривается проблема извлечения отношений из текстов на естественных языках. Отношением будем называть слово или несколько слов в предложении, соответствующие предикату в RDF-триплете.

Рассмотрим пример. Предложение *«Рихард Штраус родился в Мюнхене»* соответствует RDF-триплету (в скобках указаны соответствующие URI сущностей в DBpedia):

Субъект: Рихард Штраус (http://DBpedia.org/page/Richard_Strauss)

Предикат: место рождения (http://DBpedia.org/ontology/birthPlace)

Объект: Мюнхен (http://DBpedia.org/resource/Munich)

В предложении предикат «место рождения» описан при помощи слов «родился в». Можно переформулировать данное предложение: «Местом рождения Рихарда Штрауса считается Мюнхен». Соответствующий триплет остался неизменным, но теперь тот же самый пре-

дикат описывается словами «считается местом рождения». Благодаря особенностям естественных языков один и тот же предикат можно описать десятками различных способов даже в рамках одного языка. Извлечение отношений из текстов является одной из важных задач в области обработки естественных языков. Возможность извлекать отношения позволяет структурировать тексты и использовать из для дальнейшей машинной обработки. Кроме того, автоматическое извлечение отношений является оптимальным способом обогащения онтологий. Еще одна важная задача, которую может решить извлечение отношений, - улучшение структуры онтологий. В частности, одна из проблем при извлечении информации из Википедии состоит в том, что одни и те же понятия могут быть выражены в шаблонах разными способами, например, понятие «место рождения» может быть сформулировано в английском языке как «birthplace» и как «placeofbirth». При построении DBpedia этот аспект учитывается, однако там встречаются случаи, когда несколько предикатов имеют один и тот же смысл. Например, предикаты, соответствующие URI:

- http://DBpedia.org/property/birthDat
- http://DBpedia.org/property/birthdat
- http://DBpedia.org/property/birthdate
- http://DBpedia.org/property/birthDate
- http://DBpedia.org/property/birtdate
- http://DBpedia.org/property/birthBirth

- все связывают субъект с датой его рождения, то есть для их описания было бы достаточно единственного предиката.

Наличие таких неточностей в структуре онтологии усложняет ее использование: при использовании такого отношения в запросе нужно учитывать все возможные его варианты, представленные в онтологии. Автоматическое извлечение заранее заданных отношений из текстов позволило бы сравнить словосочетания, используемые для описания разных предикатов и сделать вывод о том, что это один и тот же предикат.

Таким образом целью работы стало исследование существующих методов извлечения отношений из текстов на естественных языках, построение метода извлечения отношений для русского языка на основе базы знаний DBpedia, его реализация и оценка качества.

1 Постановка задачи

Целью данной работы является исследование и разработка методов извлечения заданных отношений из текстов на русском языке с использованием онтологии проекта DBpedia.

Для достижения указанной цели были поставлены следующие задачи:

1. Исследовать существующие методы извлечения отношений из текстов на естественных языках, в том числе методы, использующие базы знаний и онтологии.

2. Разработать алгоритм извлечения отношений из текстов на русском языке на основе DBpedia.

3. Создать систему извлечения отношений из текстов на основе DBpedia, подтверждающую работоспособность разработанного метода. Произвести экспериментальные исследования построенной системы.

2 Обзор существующих решений

В данном разделе будут рассмотрены основные существующие методы извлечения отношений из текстов на естественных языках в п. 2.1 – 2.5. Пункт 2.6 содержит заключение.

Определим критерии, на которые будем обращать внимание при сравнении рассматриваемых методов извлечения отношений:

1. Язык. Возможно ли использовать метод для извлечения отношений из текстов на русском языке без кардинальных изменений. Все рассматриваемые методы разрабатывались для обработки текстов на английском языке, однако часть предложений, приводимых в качестве примера, в данном обзоре переведены на русский с сохранением важных семантических и синтаксических особенностей в том случае, когда перевод необходим для понимания работы метода.
2. Универсальность по отношениям — для всех ли предикатов применим рассматриваемый метод. Методы, успешно работающие для одних предикатов, могут показывать плохой результат для других.
3. Формат результата - возможность интеграции с существующими онтологиями и использование.
4. Точность и полнота. При рассмотрении этого аспекта следует учитывать, что методы тестировались на различных данных и разрабатывались для решения различных задач, поэтому использовать этот критерий для сравнения методов между собой было бы некорректно. Однако полезно учитывать эти характеристики как показатель эффективности решения поставленной авторами метода задачи.
5. Отдельное внимание уделим задаче, которую решает рассматриваемый метод.

Анализ существующих решений по указанным параметрам позволит выбрать нужную стратегию для решения поставленной задачи.

Среди известных автору работы систем извлечения отношений с русским языком работает только разработанный компанией Яндекс Томита-парсер[3]. Эта система позволяет извлекать отношения из текстов на естественных языках при помощи контекстно-свободных грамматик и словарей ключевых слов. Для каждого отношения, которое нужно уметь извлекать, необходимо описывать грамматику и отдельные словари, что делает данный метод неэффективным для решения поставленной задачи: в DBpedia содержится несколько

[3] http://api.yandex.ru/tomita/

тысяч отношений и их количество растет. Необходимость ручного задания правил для извлечения каждого отношения делает систему плохо масштабируемой. Поэтому Томита-парсер не будет рассматриваться в работе как один из возможных методов решения поставленной задачи.

Перейдем к рассмотрению существующих методов извлечения отношений из текстов на естественных языках.

2.1 Snowball

Система **Snowball** [5] была разработана в 1999 году на основе метода DIPRE[6] (Dual Iterative Pattern Expansion — двух-шаговое построение шаблонов), предложенного С. Брином для извлечения отношений из коллекций HTML-документов.

Отношениями в данной работе называются кортежи <организация, локация> (например, <Microsoft, Redmond>).

Идея метода основывается на предположении, что субъект и объект, образовывающие искомый кортеж, встречаются в схожем контексте в разных текстах коллекции документов.

Метод DIPRE был предложен одним из первых, однако практически не использовался в чистом виде, поэтому мы рассматриваем его модификацию, реализованную в Snowball.

Извлечение отношений происходит в несколько итераций:

1. Задается начальный набор кортежей.
2. Производится поиск предложений, в которые входят субъекты и объекты из набора в п.1
3. Анализируется контекст. В результате этого шага строятся шаблоны. Шаблон в Snowball представляет собой набор из пяти элементов: помимо субъекта и объекта в шаблон попадают слова или группы слов, предшествующих началу кортежа, лежащих между объектом и субъектом, либо сразу после кортежа.

   Например: <organization> based in <location>, где organization и location – тэги именованных сущностей, в отличие от классического метода DIPRE, который использовал в шаблонах регулярные выражения.
4. При помощи полученных шаблонов извлекаются новые кортежи и новый набор подается на вход в п.2.

Рисунок 2: Схема работы Snowball

Для отсева лишних кортежей каждому элементу шаблона назначается вес в зависимости от количества вхождений. При переходе на новую итерацию шаблоны с весом ниже порогового отсекаются.

Для тестирования авторы использовали корпус новостных статей North American News Text Corpus[4]. Для тренировки (построения шаблонов) использовался набор данных из 178 000 статей, для тестирования — 142 000 документов.

Тестирование проводилось для различных значений весов элементов шаблонов, максимальная полнота составила 80%, максимальная точность — 72%.

Основной целью разработчиков данного метода была максимальная автоматизация — экспертные действия свелись к заданию начального набора кортежей. Среди минусов можно отметить извлечение только одного типа отношений.

Данный метод разрабатывался для английского языка и плохо применим к русскому, поскольку полученные шаблоны зависят от порядка слов в предложении: в русском языке, в отличие от английского, порядок слов в предложении не фиксирован.

2.2 NELL

NELL [7] (Never-ending Language Learning – бесконечное изучение языка) — система извлечения отношений, предложенная в 2010 году. Основной целью разработчиков было создание системы, не только непрерывно обрабатывающей тексты и обогащающей полученной из них информацией базу знаний, но и способной улучшать качество извлечения информации. Под «бесконечно изучающей» авторы алгоритма подразумевают систему, ежедневно выполняющую две задачи:

[4] http://www.ldc.upenn.edu

1. *Чтение*: извлечение информации из текстов для дальнейшей структуризации и расширения базы знаний.
2. *Обучение*: увеличение качества чтения каждый день, как следствие возможности обработать повторно вчерашние тексты, извлекая из них больше информации и с большей точностью.

Отношения в NELL разделяются на две категории:

1. *Отношения-кандидаты* — отношения, которые пока нельзя считать достоверными фактами.
2. *Достоверные факты* — кандидаты, которые набрали достаточный вес, чтобы войти в онтологию.

Систему, предлагаемую авторами NELL, можно естественным образом разделить на следующие компоненты:

1. *База знаний* — онтология, хранящая все известные системе на данный момент факты. На момент запуска в ней должны быть описаны все используемые предикаты и задан некоторый стартовый набор отношений. В дальнейшем набор отношений увеличивается, в этом и состоит задача системы. Набор же предикатов остается неизменным.
2. *Подсистема компонентов извлечения\обучения*, которые предлагают новые возможные отношения на основе текущего состояния базы знаний и данных из внешних источников (п.3). Модули подсистемы определяют для каждого отношения-кандидата вероятность того, что оно достоверно, зависящую от надежности источника.
3. *Внешние источники* — коллекции текстов, Интернет.
4. *Модуль интеграции знаний* — проверяет отношения-кандидаты и те из них, которые имеют наибольший вес, присвоенный модулями п.2, переходят в категорию достоверных фактов и добавляются в онтологию.

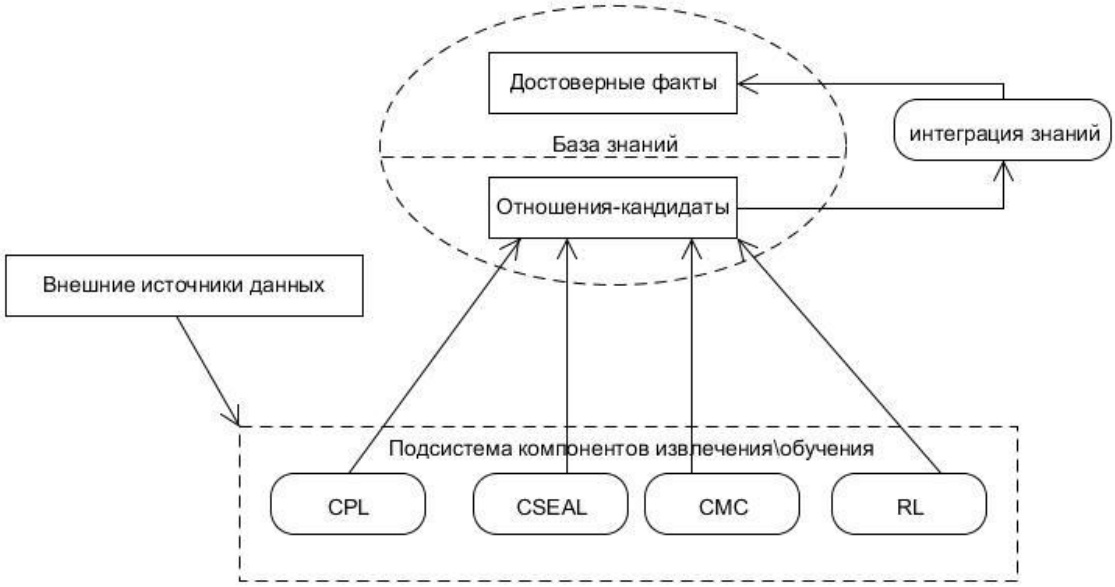

Рисунок 3: Структура системы NELL

Система работает итеративно, на каждой итерации база знаний расширяется новыми отношениями и модули подсистемы переобучаются.

Существует вероятность накопления ошибок системой: один ошибочно добавленный в базу факт приведет к извлечению новых ошибочных же фактов. Во избежание этого авторами проекта было решено, что 15 минут в день системой занимается эксперт, удаляя ошибочные факты. Кроме того, при разработке старались не использовать компоненты, которые генерируют коррелирующие ошибки. Чем ниже вероятность, что все модули допустят ошибку на одном и том же отношении — тем ниже вероятность добавить ошибочный факт в онтологию.

Наибольший интерес в рамках данного обзора для нас представляют модули, отвечающие за предложение новых отношений. Подсистема, упомянутая в п.2 и реализованная авторами NELL, состоит из следующих компонентов:

1. ***Coupled Pattern Learner*** (CPL) [8] – модуль, извлекающих отношения из текста на основе контекстных шаблонов (например, «X — президент У» или «X играет за У»). Шаблоны строятся отдельно для каждого интересующего предиката, на основе разметки по частям речи и частоты употребления каждого сочетания тэгов. Слишком общие шаблоны (которые подходят для многих предикатов) – отсеиваются, остальные используются для поиска новых отношений, соответствующих рассматриваемому предикату. Авторы работы использовали средства пакета OpenNLP[5] для

[5] `http://opennlp.sourceforge.net`

токенизации и разметки английской части корпуса ClueWeb09 [9], содержащей около 500 млн веб-страниц.

2. ***CoupledSEAL*** (CSEAL) [8] – модуль, запрашивающий объекты известных отношений в поисковых системах и извлекающий из результатов запроса отношения-кандидаты. Например, для запроса «Ford, Nissan, Toyota» модуль CSEAL возвращает набор других марок автомобилей: «Audi, BMW, Honda, …».
3. ***Coupled Morphological Classifier*** (CMC) – набор модулей логистической регрессии (по одному на каждое отношение), которые используют достоверные факты из базы знаний как тренировочные сущности и распознают отношения по совокупности морфологических признаков (большие буквы, аффиксы, части речи и проч.). Модуль работает только для предикатов, имеющих не менее 100 соответствующих фактов в базе знаний. Этот модуль, как и CSEAL, использует взаимоисключающие отношения для генерации списка исключений, по которому фильтруется результат работы системы, что значительно повышает точность.
4. ***Rule Learner*** (RL) — модуль, реализующий почти в точности алгоритм FOIL [10] и обучающийся извлекать отношения на основе дизъюнктов Хорна [11]. Дизъюнкт Хорна – это дизъюнкция литералов с не более чем одним положительным литералом (например, $\neg a \vee \neg b \vee c$). При помощи дизъюнктов Хорна описываются правила, по которым отношения соотносятся между собой.

Реализованная система расширила начальную онтологию из 123 сущностей и 55 отношений 242,453 новыми фактами с точностью 74% за 67 дней работы. Наибольшее количество отношений было извлечено с использованием модуля CPL (как в одиночку, так и в совокупности с другими модулями).

Данный метод не может быть использован для русского языка без изменений, т.к. некоторые из описанных модулей (CPL) основываются на строении предложений в английском языке. Однако в целом система NELL решает задачу, схожую поставленной в дипломной работе, поэтому рассмотренный метод будет одним из опорных при построении решения.

2.3 OpenIE 3.0 (OLLIE)

OpenIE[6] (Open Information Extraction – открытое извлечение информации) [12] – проект с открытым кодом[7], который занимается разработками методов извлечения информации. На

[6] http://openie.cs.washington.edu/

момент написания дипломной работы представлен код версии OpenIE 4.0, однако отсутствуют статьи, описывающие экспериментальные результаты и характеристики новой системы. Поэтому в данной работе рассматривается предыдущая версия проекта – OpenIE 3.0, известная так же как **OLLIE** (Open Language Learning for Information Extraction – открытое изучения языка для извлечения информации).

Общая отличительная черта всех систем проекта OpenIE - извлечение отношений из текста без использования начальных наборов отношений и информации о предметной области.

Система OLLIE позволяет формировать из слов входного предложения триплеты, выражающие отношение между двумя аргументами в виде (аргумент1; отношение; аргумент2).

Для тренировки OLLIE использовали отношения, извлеченные предыдущей версией OpenIE – ReVerb [13]. В отличие от ReVerb, OLLIE извлекает не только отношения, ключевое слово в которых представлено глаголом, но и отношения, предикат в которых описывается существительными. Кроме того, OLLIE извлекает отношения с учетом контекста, чего не было в предыдущих версиях OpenIE. Однако, ReVerb имеет достаточно хорошую точность, чтобы извлеченный им набор отношений целесообразно было использовать для обучения OLLIE. На этом наборе OLLIE строит *открытые шаблоны.* Далее эти шаблоны применяются для извлечения информации из текстов. OLLIE анализирует контекст, чтобы получить дополнительную информацию об извлеченных триплетах. Дополнительной информацией может быть, например, степень достоверности факта: предложения «*Мартин Генрих Клапрот первым получил уран*» и «*Говорят, что Мартин Генрих Клапрот первым получил уран*» описывали одно и то же отношение для ReVerb, но разные для OLLIE.

[7] https://github.com/knowitall/ollie

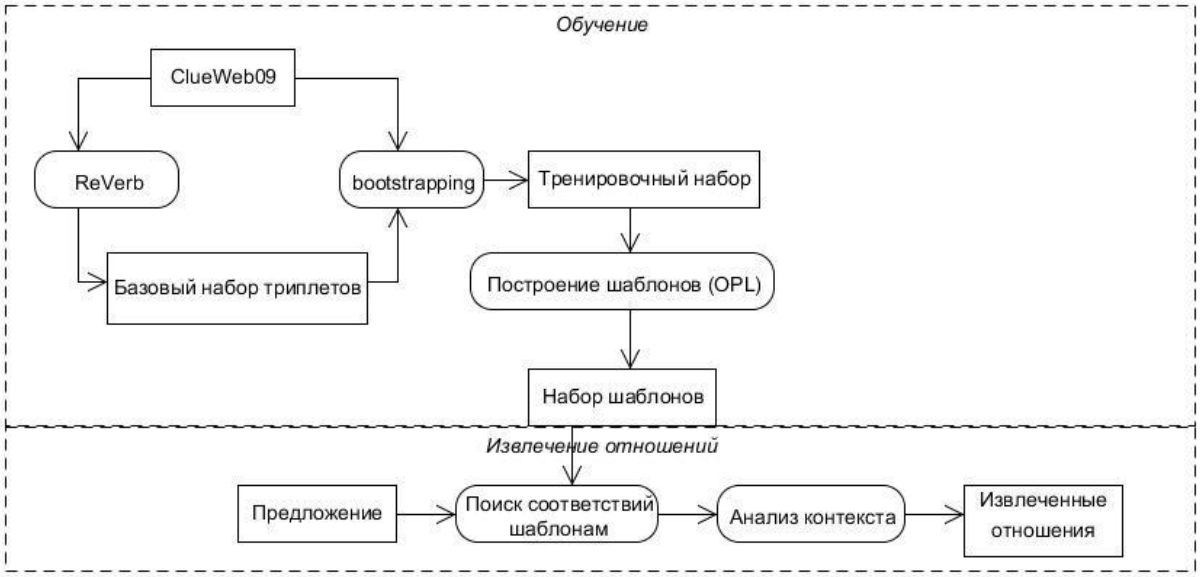

Рисунок 4: Схема работы OLLIE.

Рассмотрим подробнее этапы работы системы.

1. ***Генерация обучающей выборки***. Среди отношений, извлеченных при помощи ReVerb из корпуса ClueWeb09, отобрали 110 000 таких, которые встретились не менее двух раз и аргументами в которых являлись имена собственные. Далее в том же корпусе были найдены все предложения, содержащие аргументы полученного набора отношений. Авторами была предложена следующая гипотеза: все эти предложения описывают ту же информацию, что и полученные ReVerb триплеты. Однако, это выполняется не всегда: для пары аргументов <Бойль, Австрия> в корпусе можно найти предложения «*Бойль родился в Австрии*» и «*Феликс родился в маленьком городе на севере Австрии, где Бойль ходил в школу*». Очевидно, что первое предложение описывает отношение <Бойль; место рождения; Австрия>, а второе – два отношения: <Феликс; место рождения; Австрия> и <Бойль; место обучения; Австрия>. Для того, чтобы минимизировать количество подобных ошибок, были добавлены дополнительные ограничения на искомые предложения: в графе зависимостей предложения между искомыми словами должен быть путь длиной не более 4. С учетом этого ограничения количество найденных предложений уменьшилось с 18×10^6 до 4×10^6 предложений. Авторы проекта так же рассматривали построение сложной вероятностной модели как альтернативу описанному ограничению, но пришли к выводу, что тренировочный набор, построенный используемым методом, имеет более высокое качество и лучше подходит для обучения системы.

2. ***Open Pattern Learning*** (построение открытых шаблонов). На следующем шаге необходимо научиться фиксировать шаблоны, описывающие структуры, кодирующие отношения в естественном языке. Шаблон фиксирует

   - порядок аргументов
   - их синтаксические зависимости
   - часть речи, соответствующую предикату
   - может включать предлоги и слова, общие для определенного типа отношений.

   Например, отношению (arg1; be {rel} of; arg2) соответствует открытый шаблон {rel:postag=NN; type=Person} ↑nn↑ {arg1} ↑nn↑ {arg2}, где ↑nn↑ обозначает связь в дереве зависимостей. Открытый шаблон фиксирует, каким образом отношение может быть описано в тексте при помощи пути зависимостей. Шаблоны, подходящие для многих отношений сразу – удаляются. Открытые шаблоны разделяются на *чисто синтаксические* и *семантические* при проверке выполнения ряда условий, ограничивающих структуру шаблона (например, узловое слово для отношения расположено внутри аргумента): если условия выполнены, то полученный шаблон - чисто синтаксический и для него проверка заканчивается, шаблон готов к использованию. Если условия не выполнены, то рассматриваемый шаблон считается семантическим. Семантические шаблоны встречаются не так часто, как синтаксические, однако достаточно для того, чтобы их неправильное использование влияло на точность работы системы.

   Для иллюстрации проблемы обработки семантических шаблонов сравним фразы «Microsoft co-founder Bill Gates» и «Chicago Symphony Orchestra» – несмотря на то, что они имеют синтаксически одинаковую структуру, одинаково обрабатывать их нельзя. Используя для них один открытый шаблон получим: <Bill Gates; co-founder; Microsoft> (Билл Гейтс – сооснователь Microsoft) и <orchestra; symphony; Chicago> (оркестр — симфония Чикаго). В таких случаях необходим анализ контекста. Авторы метода предлагают обобщать использование шаблонов при помощи списков-категорий: первый из рассматриваемых в примере шаблонов успешно работает для отношений «директор», «основатель», «начальник» и подобных им.

   Дальше полученные шаблоны упорядочиваются в порядке убывания частоты их появления: предполагается, что шаблон, который чаще всего встречается в тренировочном корпусе, даст наибольшую точность извлечения.

3. ***Извлечение отношений при помощи открытых шаблонов***. Открытый шаблон накладывается на дерево зависимостей предложения в поисках узлов, которые являются потенциальными аргументами отношения. Далее происходит анализ контекста во избежание потери важной информации. В том числе в аргументы добавляются слова, входящие в группы соответствующих узлов если это группы существительных.

4. ***Анализ контекста***. Обработка контекста в OLLIE использована главным образом для обработки отношений, для которых в тексте указана их возможная достоверность. Большинство таких предложений содержат фиксированные вводные конструкции и обрабатываются OLLIE: система добавляет дополнительное поле в кортеж и некорректное отношение становится корректным. Например, из предложения «*Кэри сказала, что температура плавления льда – 15 градусов*» вместо отношения *<лед; температура плавления; 15 градусов>* будет извлечено отношение *<Кэри; считает; <лед; температура плавления; 15 градусов>>*. Обрабатываются условные обороты при помощи дерева зависимостей и стоп-слов.

Экспериментальные исследования показали, что каждый из этапов обработки значительно влияет на эффективность работы системы. Так, например, контекстный анализ повышает точность на 20% (при максимальной точности системы 80%). Полнота результата не замерялась. При анализе ошибок системы было установлено, что наибольшая доля ошибочных извлечений происходит из-за некорректной работы парсеров, отвечающих за построение дерева зависимостей.

Данная система разрабатывалась для английского языка и не может быть использована для текстов на русском языке без переобучения (что не является тривиальной задачей, т.к. ReVerb, предоставляющий начальный набор отношений для обучения, также не работает с русским языком) и изменения алгоритма анализа контекста. Положительным качеством системы является ее автономность (участие человека в работе системы не требуется), в отличие от рассмотренных ранее в обзоре методов.

2.4 SOFIE

SOFIE [14] (Self-organizing framework for information extraction – самоуправляемая среда для извлечения информации) разрабатывалась как система автоматического обогащения онтологий.

В качестве иллюстрации процесса работы системы, авторы приводят примерный сценарий.

Предположим, что на вход системе извлечения фактов приходит предложение «*Эйнштейн закончил школу в Германии*». Зная, что Эйнштейн — это фамилия ученого Альберта Эйнштейна и что Альберт Эйнштейн родился в Германии, система может сделать вывод, что «*X закончил школу в Y*» - отношение, соответствующее предикату «место рождения».

Предположим, что далее на вход системе пришло предложение «Элвис закончил школу в Мемфисе». Элвис — достаточно распространенное имя, но пусть из контекста следует, что речь идет об Элвисе Пресли. Однако из базы знаний известно, что Элвис Пресли родился в штате Миссисипи. В системе записано, что у одного человека может быть только одно место рождения. Получили противоречие. Значит, шаблон «закончил школу в» не может описывать предикат «место рождения».

Тогда система снова возвращается к первому предложению и обнаруживает из контекста, что речь идет об отце Эйнштейна — Германе, который связан с Германией отношением «получил образование в». Система делает вывод, что полученный шаблон соответствует предикату «получил образование в», и соответственно из второго предложения извлекается триплет <Элвис Пресли; получил образование в; Мемфис>.

Таким образом, задача разделяется на три подзадачи:

1) построение шаблонов, с использованием которых будут извлекаться отношения из текстов;
2) разрешение лексической неоднозначности – для интеграции с онтологией необходимо различать город Пушкин и поэта Пушкина, т.к. они являются разными сущностями и имеют разные идентификаторы, а кроме того могут иметь различные ограничения на возможные отношения (у города может быть мэр, а у поэта - нет);
3) проверка целостности – новые отношения, добавляемые в онтологию, не должны противоречить уже имеющимся там.

Однако авторы SOFIE предлагают не разделять решение этих подзадач и предлагают следующую модель данных.

Модель SOFIE

Модель SOFIE можно условно разделить на две части – высказывания и правила.

Высказывание (statement) представляет собой кортеж – отношение и связанные им объекты и коэффициент истинности (truth value), равный 0 или 1. Высказывание *bornIn(AlbertEinstein, Ulm)[1]* означает, что Альберт Эйнштейн родился в Ульме и это достоверный факт.

Фактами называются высказывания, коэффициент истинности которых равен 1. Высказывания, коэффициент истинности которых неизвестен, называются *гипотезами*.

SOFIE может работать с любой онтологией, данные которой можно представить как совокупность фактов. Авторы метода для экспериментальных исследований использовали онтологию YAGO[15].

Одна из проблем SOFIE – слова, имеющие несколько значений. Для работы с различными значениями одного слова вводится обозначение *wic* (word in context – слово в контексте). Контекстом считается документ, в котором встретилось слово, поэтому *wic* представляет собой пару <слово>@<идентификатор документа>. Например, Java@D3. Предполагается, что одинаковые *wic* имеют одинаковое значение.

Вся информация, с которой работает SOFIE представляется в виде высказываний. В том числе:

- использование шаблона в тексте - *patternOcc("X went to school in Y", Einstein@D29, Germany@D29)[1]* – шаблон «*X went to school in Y*» связывает объекты Einstein и Germany в том значении, в котором они использованы в документе D29;
- разрешение неоднозначности - *disambPrior(Elvis@D29, ElvisPresley, 0.8)[1]* – упомянутый в документе D29 Elvis с вероятностью 0.8 имеет значение ElvisPresley.

Основываясь на фактах, извлеченных из текстов и фактах, уже имеющихся в онтологии, SOFIE строит гипотезы. Гипотезы могут предполагать разрешение неоднозначности - *disambiguateAs(Java@D8, JavaProgrammingLanguage)[?]*, соответствие шаблона определенному отношению - *expresses("X was born in Y", bornInLocation)[?]* или связь двух аргументов отношением - *developed(Microsoft, JavaProgrammingLanguage)[?]*.

Для проверки гипотез SOFIE нужны *правила*.

Правила ограничения на разрешение гипотез. Например, правило, контролирующее факт наличия не более одного места рождения для одного человека, может записываться следующим образом:

$bornIn(X, Y) \& \neg sameAs(Y, Z) \Rightarrow \neg\, bornIn(X, Z)$

Каждое правило имеет свой вес, влияющий на разрешение гипотезы: некоторые правила могут только снижать коэффициент истинности при их невыполнении, другие – устанавливать его в 0, и тогда гипотеза сразу считается неверной.

Рассмотрим алгоритм построения шаблонов в SOFIE.

Первый шаг построения - токенизация входного документа (разбиение его на короткие строки – токены). На данном этапе происходит обнаружение и нормализация чисел, дат, распознавание гиперссылок в статьях Википедии. Система использует составленные экспертами списки для определения стоп-слов, слов, описывающих национальности, профессии и др. Определяются токены, которые должны обозначать людей.

Следующий шаг — определить какие именно токены интересны для дальнейшей работы. Таковыми считаются даты, числа и имена собственные. Как только два интересных токена оказываются на определенном расстоянии друг от друга — система генерирует высказывание «встретился шаблон» (pattern occurrence fact): *patternOcc(p, x@d, y@d)[1]*. Оптимальное расстояние между токенами было подобрано экспериментально и равно

Далее для полученной гипотезы вычисляется ее правильность с использованием известных фактов и правил.

Данная система была протестирована на нескольких различных корпусах и показала разную точность для разных видов отношений (от 80% для отношения *directed* до 100% для отношения *ownedBy*).

На тестовом корпусе, использованном для тестирования Snowball (п. 2.1), система SOFIE показала точность - 91.43% и полноту - 24.32%.

2. 5 Система обогащения DBpedia

P. Nugues и P. Exner в [16] предлагают систему, обеспечивающую полный цикл извлечения отношений: от неструктурированного текста до RDF-триплетов. С развитием концепции Семантической паутины проблема интеграции систем извлечения информации с существующими онтологиями становится все более актуальной. Основной задачей разработки было построение системы, позволяющей обогащать онтологию DBpedia с использованием данных из текстов на естественном языке.

Баланс между точностью и полнотой обеспечивается за счет использования комбинации NLP средств, включающих анализ семантической структуры, разрешение кореферентно-

сти, связывание именованных сущностей. Расширяемость достигается благодаря параллельной работе модулей системы на кластере.

Особенность предложенного метода в том, что не создается каких-либо новых семантических структур: работа идет сразу с идентификаторами DBpedia, что упрощает задачу ее обогащения.

На рисунке ниже представлена схема работы системы.

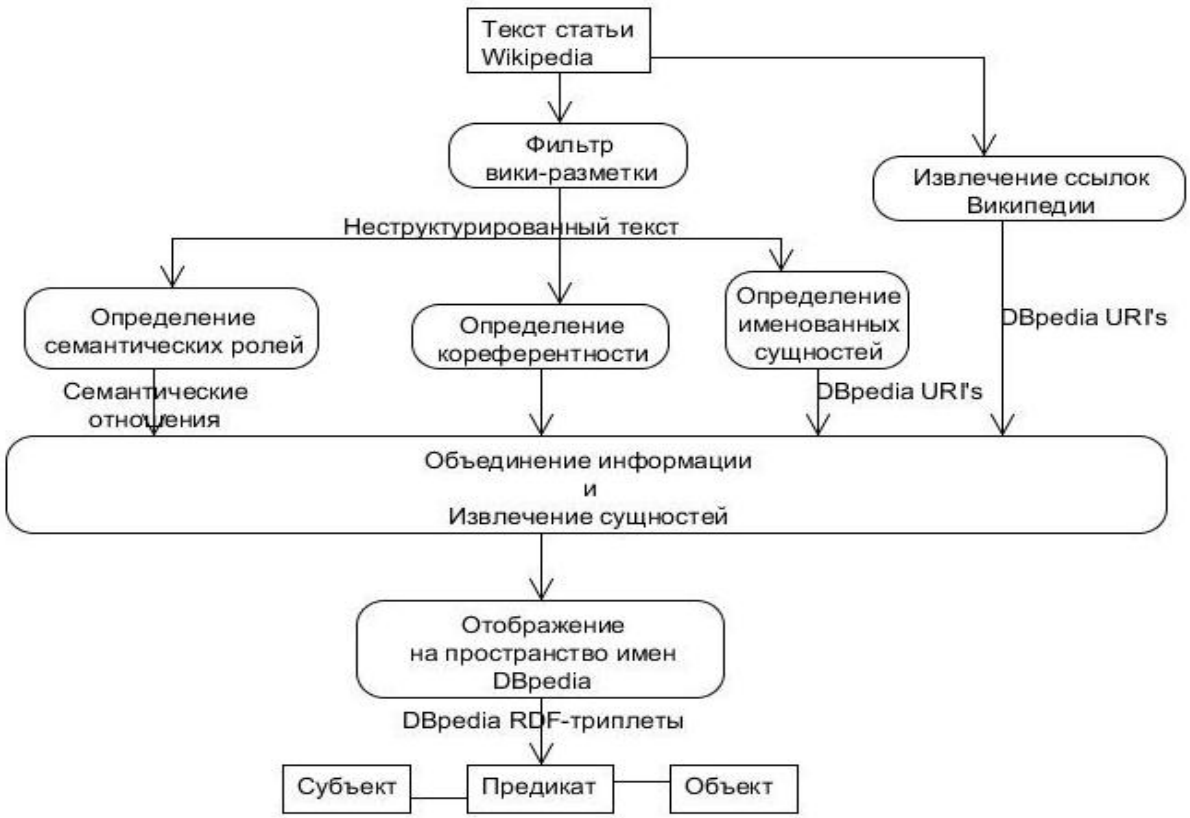

Рисунок 5: Схема работы системы обогащения DBpedia

Построенная система представляет собой последовательную работу 7 модулей обработки текста:

1. ***Фильтр вики-разметки*** преобразует статьи Википедии в чистый текст. На вход системе подаются статьи из Википедии, которые помимо текста на естественном языке содержат аннотации и специальную разметку, называемую вики-разметкой. Для дальнейшей обработки необходимо выделить из статьи только текст, чем и занимается данный модуль. Удаляется аннотация и разделы, состоящие только из ссылок. Ошибки на первом этапе часто возникают из-за незакрытых или неправильно вложенных HTML-тегов.

2. Модуль ***извлечения ссылок*** определяет, на какие статьи Википедии ссылается обрабатываемый текст. Указанные в тексте статьи ссылки сохраняются с соответствующими им фрагментами текста. Кроме того, делается предположение, что первая группа существительного в первом предложении соответствует ссылке на статью. Это предположение вызвано тем, что наиболее длинные цепочки кореферентности как правило начинаются с первого упоминания сущности в тексте. Кроме того, взаимно-однозначное соответствие ресурсов Википедии и DBpedia позволяет определить URI для объектов, на которые указывают ссылки статьи.
3. Модуль ***семантической разметки*** определяет грамматическую информацию для слов в тексте. Структурная семантика (frame semantics, [19]) – лингвистическая теория, утверждающая, что значение предложения можно представить как набор предикатов и аргументов. The Proposition Bank (Propbank, [20]) – проект, в котором эта теория использовалась для разметки корпуса предикатами и аргументами. Для каждого предиката Propbank позволяет определить до шести зависимых слов A0, … A5, в отличие от классической структуры объект-предикат-субъект. Для семантической разметки авторы системы использовали среду Athena[21].
4. Модуль ***разрешения кореферентности*** позволяет определить, какие имена ссылаются на один и тот же объект в тексте. Во-первых, это уменьшает количество рассматриваемых сущностей. Во-вторых, соответственно увеличивает количество потенциальных триплетов для каждой сущности. Так, в статье про Альберта Эйнштейна все отношения со словами «он», «физик», «великий ученый» и пр. будут успешно извлечены, но использовать их для обогащения онтологии нельзя. Эту проблему и устраняет модуль разрешения кореферентности. В предложенной системе использовалось средство из пакета Stanford CoreNLP[22].

   Среди существующих систем извлечения информации лишь немногие представляют полный анализ исходных документов и используют разрешение кореферентности. Исключениями, помимо рассматриваемой в текущем разделе системы, являются LODifier[17] и KnowledgeStore[18]. Однако сущности, которые извлекаются ими, не могут быть интегрированы в онтологию. В остальном эти две системы используют методы, рассмотренные выше, поэтому в данном обзоре рассматриваться не будут.

5. Модуль, ставящий каждой сущности в соответствие некоторый URI из DBpedia. В большинстве статей Википедии только первое упоминание объекта связано со ссылкой на соответствующую статью, все последующие упоминания связанных с ними ссылок уже не имеют. Для восстановления ссылок был использован Wikifier [23] – система, аннотирующая текст соответствующими ссылками на страницы в Википедии. Благодаря соответствию ссылок Википедии и URI DBpedia, Wikifier позволяет определять также URI для всех сущностей в тексте.
6. Модуль, собирающий воедино информацию от предыдущих модулей и выделяющий отношения в форме триплетов. Как правило, аргумент A0 становится субъектом отношения, а A1 – объектом. При этом и объект и субъект должны быть выражены именованной сущностью – для них должна быть определена ссылка на статью в Википедии или URI из DBpedia, либо они должны входить в цепочку разрешения кореферентности для других именованных сущностей.
7. Модуль, ставящий предикатам из полученных триплетов в соответствие отношения из DBpedia. На этом этапе имеется набор отношений, предикаты в которых описаны в соответствии со словарем Propbank, содержащим около 20 000 возможных предикатов. DBpedia содержит около 1600 предикатов. Необходимо найти соответствие между двумя системами описания предикатов. Это нетрудно сделать, сравнивая предикат Propbank с предикатом DBpedia для уже имеющихся в онтологии отношений.

Рассмотрим пример обработки предложения.

«*Люк Бессон (родился 18 марта 1959) – французский режиссер, писатель и продюсер. Бессон родился в Париже.*»

Модуль 3 определяет два потенциальных триплета: «*Люк Бессон - родился - 18 марта 1959*» и «*Бессон – родился – в Париже*».

Модуль 4 определяет, что имена «Люк Бессон» и «Бессон» называют одного человека. Модуль 5 определяет для субъекта «Люк Бессон» и объектов URI, соответствующие им в DBpedia.

Модуль 6 собирает воедино информацию от предыдущих модулей и находит в DBpedia отношения, соответствующие заданным парам субъект-объект. Для первого триплета это <DBpedia-owl: birthDate>, для второго - <DBpedia:birthPlace>. Каждому полученному отношению в соответствие ставится предикат «родился».

Если в дальнейшем в тексте встретится триплет с предикатом «родился» – появилась возможность определить отношение между субъектом и объектом, даже если оно отсутствует в DBpedia. И более того – добавить его в DBpedia.

В рассматриваемой работе обрабатываются только триплеты, предикаты которых описывают время или расположение (AM-TMP и AM-LOC в Propbank). Для извлечения дат использовались шаблоны, к объектам местонахождения относились сущности, соответствующие классу <DBpedia:Place> в онтологии DBpedia.

Представленная система была протестирована на 114 000 статей Википедии. В результате смогли извлечь более 1 000 000 отношений. Для определения качества использовались 200 случайных статей из Википедии, размеченных вручную.

F-мера составила 66.3%, точность – 74.3%, полнота – 59.9%.

Данная система не может быть использована для русского языка, т.к. русский язык не поддерживается модулями обработки текстов, из которых она состоит.

2.6 Сравнение рассмотренных методов

Были рассмотрены основные методы извлечения отношений из текстов на естественном языке. Можно заметить, что большинство из них используют построение шаблонов для извлечения отношений. В таблице ниже приведено сравнение методов по параметрам, описанным в начале обзора:

- Система – название системы (для системы из п.2.5 использовали фамилии авторов, т.к. предложенная ими система названия не имеет);
- Русский язык – возможность использования системы для текстов на русском языке;
- Отношения – для каких отношений работает рассматриваемая система;
- Ручная работа – необходимость участия экспертов в работе системы;
- Онтология – возможность интеграции с существующими онтологиями.

Таблица 1: Сравнение существующих систем извлечения отношений

Система	**Русский язык**	**Отношения**	**Ручная работа**	**Онтология**
Snowball	Нет	Только <organization, location>	Начальный набор кортежей	Нет
NELL	Нет	Все, присутствующие в онтологии на момент старта	Эксперт удаляет ошибочные отношения	Да
OpenIE	Нет	Любые, без привязки к предикатам	Нет	Нет
SOFIE	Нет	Любые, описанные в модели данных	Списки сущностей и стоп-слов	Да
P. Nugues & P. Exner	Нет	Любые, имеющиеся в DBpedia	Нет	Да

Для решения поставленной задачи разрабатываемый метод должен быть интегрируем с DBpedia, работать для любого предиката из базы знаний, быть по возможности автоматическим (работать без наблюдения).

По результатам обзора существующих методов извлечения отношений из текстов на естественном языке, было решено строить модульную систему, использующую отношения из DBpedia для построения шаблонов, кодирующих отношения в тексте на русском языке.

3 Исследование и построение решения

Как следует из обзора, ни один из рассмотренных методов не удовлетворяет постановке задачи в полной мере.

Задача разрабатываемой системы – находить в тексте на русском языке отношения, описывающие имеющиеся в DBpedia предикаты, и извлекать их в виде, пригодном для дальнейшего добавления в DBpedia.

3.1 Общая схема процесса

Основные этапы работы системы описаны на диаграмме ниже.

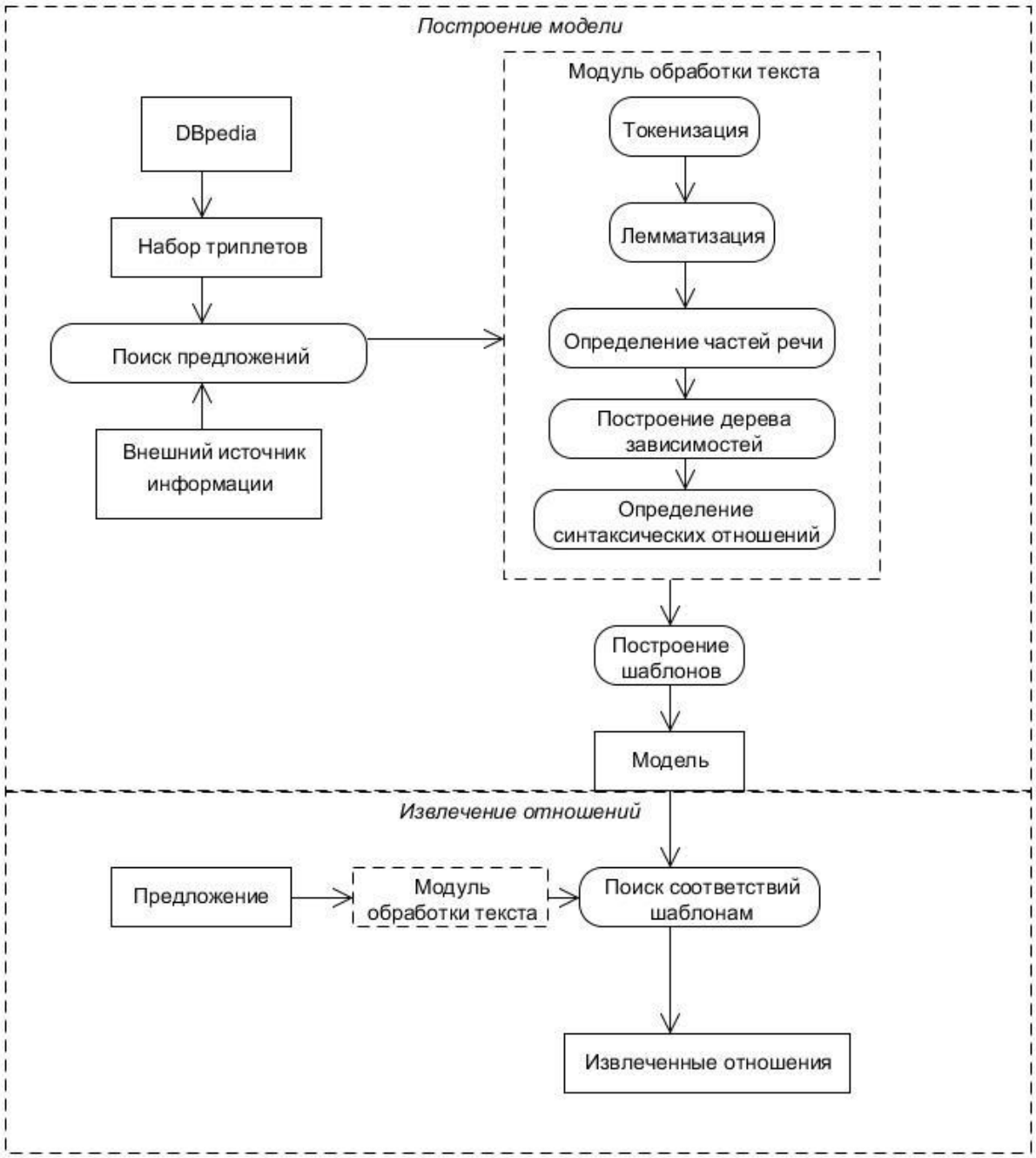

Рисунок 6: схема работы системы

Построение системы естественным образом разбивается на две подзадачи: подготовка входных данных для модели и построение модели.

В разделах 3.1.1 - 3.1.2 остановимся подробнее на каждой подзадаче. В разделе 3.1.3 рассмотрим алгоритм извлечения отношений из текста на основе построенной модели.

3.1.1 Подготовка входных данных для построения модели

На этом этапе необходимо задать предикат, для которого будет строиться модель. **Модель** в данной дипломной работе представляет собой структуру, ставящую в соответствие предикату набор описывающих его шаблонов и частоту, с которой они встречаются. Определение шаблона будет дано позже, в разделе 3.1.2.

Далее для рассматриваемого предиката необходимо получить все RDF-триплеты из DBpedia, в которых он участвует.

На этом этапе мы имеем набор достоверных фактов из онтологии, теперь нужно получить их представление в естественном языке. Для этого решено было использовать полнотекстовый поиск по статьям Википедии: для каждого триплета производится поиск предложений, в которые входят слова, описывающие его объект и субъект.

Для построения шаблонов необходимо получить синтаксическую и семантическую информацию о найденных предложениях. Для этого проводятся следующие действия:

1. Токенизация – предложение разбивается на слова и знаки препинания.
2. Лемматизация – каждое слово приводится к начальной форме (лемме) – той, в которой мы можем видеть его в словаре. Например, для существительных это форма именительного падежа единственного числа, для глаголов – инфинитив.
3. Определение частей речи (POS-tagging).
4. Определение синтаксической зависимости – для каждого слова определяется номер слова в предложении, от которого оно зависит. Слова в предложении нумеруются с 1. Если номер главного слова – 0, то данное слово является корнем дерева зависимостей.

   Дерево зависимостей – способ представления синтаксической структуры предложения, в котором узлами выступают слова предложения, а ветви помечены именами синтаксических отношений. Каждая дуга при этом идет от главного слова к зависимому.
5. Определение синтаксических отношений. **Синтаксическое отношение** – это характеристика связи слов в предложении. Отношения связывают только слова, не словосочетания. Корпус русского языка СинТагРус насчитывает около 60 возможных синтаксических отношений. В рассматриваемом методе синтаксические отношения будут использоваться для определения веса и уточнения шаблонов.

Теперь, когда мы имеем всю информацию о представлении предиката в тексте на русском языке, можно построить модель.

3.1.2 Построение модели

Следующая подзадача разрабатываемого метода – построение шаблонов, кодирующих отношения в русском языке.

Определение шаблона зависит от цели, с которой будут извлекаться отношения. В OLLIE шаблон представлял собой регулярное выражение, описывающее грамматические признаки (части речи) и зависимости между словами, описывающими отношение в предложении.

Однако поставленная перед нами задача отличается необходимостью извлекать отношения, соответствующие заранее заданным предикатам. Поэтому помимо грамматических признаков необходимо хранить в шаблоне леммы входящих в него слов.

Таким образом, в этом и последующих разделах дипломной работы **шаблоном** будем называть набор лемм и их грамматических признаков, представленных в виде поддерева дерева зависимостей, описывающего предложение. Узлы, соответствующие объекту и субъекту триплета, так же будем сохранять в шаблоне: для последующего извлечения отношений нужны будут их грамматические характеристики. Для этих узлов будем хранить только их синтаксические отношения. На рисунке ниже представлен пример шаблона, извлеченного из предложения «Альберт Эйнштейн учился в Ульме». Синтаксические отношения размечены в соответствии с корпусом СинТагРус[8].

[8] http://www.ruscorpora.ru/instruction-syntax.html

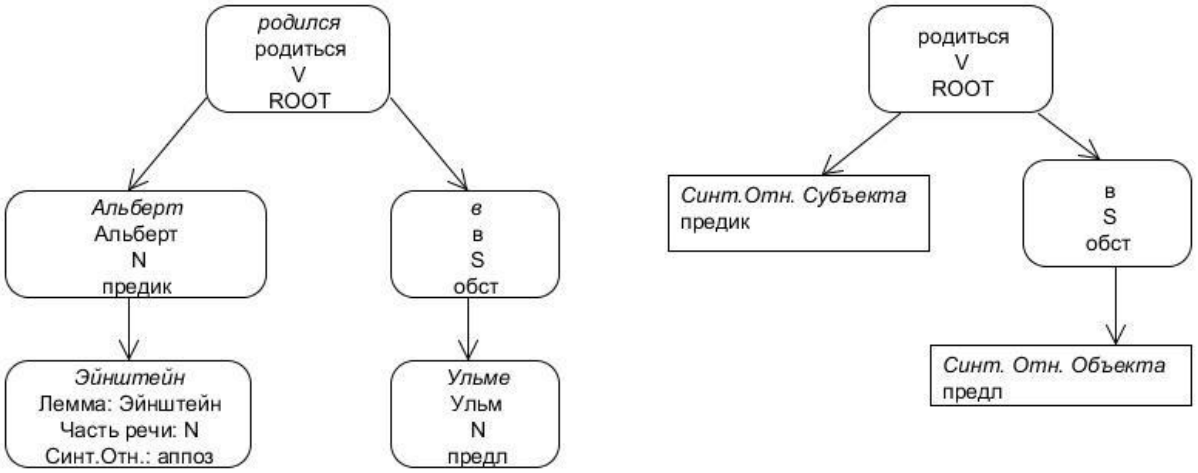

Рисунок 7. Пример извлечения шаблона. Слева – дерево зависимостей, описывающее предложение. Справа - извлеченный шаблон.

Каждому шаблону присваивается вес – вещественное число w, $0 \leq w \leq 1$. Чем больше вес – тем более достоверным можно считать полученный шаблон при извлечении отношений. После построения модели шаблоны с весами ниже экспериментально определенного порогового значения отсекаются.

Итак, имеется дерево зависимостей для каждого из набора предложений. Кроме того, известно положение объекта и субъекта в каждом из предложений. Дальнейшая разработка метода велась следующим образом:

- Выдвигается гипотеза относительно того, какие поддеревья можно считать шаблонами.
- Строится модель с использованием предложенной гипотезы.
- Предложения, использованные для построения модели, размечаются в ручную.
- Извлеченные шаблоны сравниваются с ручной разметкой входных данных – если использование гипотезы улучшает результат по сравнению с базовым методом построения шаблонов – гипотеза принимается. Иначе – отвергается.

Базовая гипотеза: любое минимальное поддерево дерева зависимостей, связывающее объект и субъект, считается шаблоном.

Построенные гипотезы и анализ результата их применения описываются в разделе 3.2.1.

Шаблоны, попавшие в несколько моделей сразу, исключаются из всех моделей, так как не позволяют однозначно определить описываемый предикат.

3.1.3 Извлечение отношений

Для извлечения отношений из предложения нужно предварительно произвести действия из п.3.1.1 для построения его дерева зависимостей.

Далее в построенном дереве нужно найти фрагменты, совпадающие с шаблонами, имеющимися в построенной в п.3.1.2 модели. Субъектом и объектом отношения становятся слова, находящиеся в вершинах, зависимых от листьев шаблона, в том случае, когда описывают синтаксические отношения, совпадающие с указанными в шаблоне.

3.2 Эксперимент

Для проведения эксперимента были выбраны следующие предикаты:

1. http://DBpedia.org/ontology/spouse - отношение супружества;
2. http://DBpedia.org/ontology/almaMater - место получения высшего образования.

Эти предикаты были отобраны по ряду причин: во-первых, они связывают между собой различные по природе сущности; во-вторых, в снимке русской DBpedia присутствует достаточно триплетов, в которых участвуют эти предикаты (например, для предиката «место рождения» в русской DBPedia триплеты отсутствуют). Кроме того, данные, содержащиеся в DBpedia несколько зашумлены и в силу этого пришлось отказаться от оценки качества на некоторых предикатах: для предиката profession (профессия, род занятий) большая часть субъектов (около 60%) имеют профессию Россия, что приводит к построению некорректной модели. Однако стоит заметить, что разрабатываемый метод не зависит от предиката, для которого извлекаются отношения, поэтому можно проводить тестирование на всех предикатах, представленных в DBpedia. Но результаты такого тестирования было бы сложно анализировать – всего в DBpedia около 1700 предикатов, среди которых присутствуют технические, связывающие онтологию с другими базами знаний, а также очень редкие и ошибочно заведенные, поэтому было решено ограничиться указанным набором предикатов.

3.2.1 Построение модели

Для каждого из рассматриваемых предикатов были получены все пары «субъект + объект», встречающиеся в триплетах с этим предикатом в DBpedia. Предложения, найденные для каждой пары, обработаны, как это описано в 3.1.1.

При поиске предложений оказалось, что требования к вхождению слов, называющих объект и субъект зависит от сущности, которую они описывают. В качестве слов, называющих объект и субъект, использовалась часть URI из DBpedia, т.к. идентификатор как правило содержит подробное название в начальной форме.

Пример: URI - http://DBpedia.org/page/Яблоко, в предложении ищется слово «яблоко».

Однако возник вопрос, как обрабатывать объекты и субъекты, описываемые в несколько слов: для людей это фамилия, имя и отчество, а также названия фильмов, компаний и проч.

Было решено, что предложение удовлетворяет требованиям, если в нем встречается хотя бы одно слово, называющее объект, и хотя бы одно слово, называющее объект.

Была предложена базовая гипотеза: шаблоном считать минимальное поддерево дерева зависимостей. Построили модель с использованием базовой гипотезы. В качестве веса использовалась частота появления соответствующего шаблона: при добавлении шаблона в модель его первоначальный вес приравнивался к 1, если шаблон еще не присутствовал в модели, и увеличивался на 1 в противоположном случае. После обработки всех предложений веса нормируются (делятся на сумму всех весов модели).

После проведения эксперимента с использованием только базовой гипотезы, построенная модель была проанализирована и выдвинуты следующие гипотезы:

1. Шаблоны, состоящие из одного слова, являющегося предлогом, в модель не добавляются, т.к. являются слишком общими и могут описывать разные предикаты. Например: «Александр [родился] в Гамбурге» и «Александр [работает] в тире» - в квадратных скобках приведены слова, опущенные в исходном предложении, но восстановимые по контексту. В первом случае предложение описывает триплет <Александр; место рождения; Гамбург>, во втором – <Александр; место работы; тир>. Без анализа контекста разделить такие случаи нельзя.
2. Если главным словом (корнем синтаксического дерева) шаблона являются слова «быть», «являться», «становиться» - в модель добавляются дополнительный шаблон без этого слова. В русском языке подобные слова могут быть опущены без изменения смысла предложения, нужно учитывать это при извлечении. Шаблоны, состоящие ровно из одного такого слова, в модель не добавляются, т.к. могут описывать различные отношения.
3. Все имена собственные исключаются из шаблонов. Оказалось, что из-за специфики результата, возвращаемого используемым API для поиска (не полноценные предложения, а сниппеты – фрагменты текста, часто несвязанного), шаблоны, в которые случайно попали имена собственные, могут набрать значительный вес и остаться в модели, что приводит к снижению точности.

В качестве примера в таблице ниже приведено влияние указанных гипотез с различными параметрами на качество построенной модели для отношения almaMater. Найденные для каждой пары объектов и субъектов предложения вручную разметили потенциальными шаблонами и сравнили с построенной моделью. Для предложения «Иван был зачислен в МГУ» верной ручной разметкой считаем два шаблона: «был зачислен» и «зачислен». Не-

корректными считаем шаблоны, которые не могут описывать рассматриваемое отношение в тексте, либо могут описывать одновременно многие отношения. Таблица описывает изменения, происходящие с моделью в порядке применения гипотез.

Таблица 2: Влияние гипотез на качество модели

Гипотеза	**Примеры шаблонов, из которых состоит модель**	**Количество шаблонов**	**Количество некорректных шаблонов в модели**
0 (Базовая)	«быть зачислить в» «поступать в» «из» «в» «окончить» «закончить» «поступить в» «быть закончить с отличие» «привести в» «Женева в» «перейти в» «князь являться»	15	8
1 (Шаблон != 1 предлог)	«быть зачислить в» «поступать в» «окончить» «закончить» «поступить в» «быть закончить с отличие»	13	5

	«привести в» «Женева в» «Альберт» «перейти в» «князь являться»		
2 (Быть, являться, становиться)	«быть зачислить в» «зачислить в» «поступать в» «окончить» «закончить» «поступить в» «быть закончить с отличие» «закончить с отличие» «привести в» «Женева в» «Альберт» «перейти в» «князь являться»	17	5
3 (Имена собственные)	«быть зачислить в» «зачислить в» «поступать в» «окончить» «закончить» «поступить в» «быть закончить с	14	3

	отличие» «закончить с отличие» «привести в» «перейти в» «князь являться»		

Для тестирования использовали набор из 200 триплетов, соответствующих отношению almaMater в DBpedia.

Кроме того, по результатам тестирования было решено удалять из модели шаблоны, имеющие вес, меньше 0.15 – для обоих рассматриваемых отношений этого порога хватило, чтобы отсечь большую часть некорректных и оставить в модели все нужные шаблоны. В зависимости от решаемой с помощью предложенного метода задачи, порог можно варьировать: меньший порог приводит к увеличению полноты, но вероятному снижению точности, и наоборот.

3.2.3 Извлечение отношений

Для тестирования качества извлечения отношений был использован искусственно созданный корпус на основе Википедия: из статей, описывающих людей, были отобраны 300 предложений и размечены отношениями almaMater и spouse вручную. Результаты тестирования приведены в таблице ниже.

Таблица 3: Точность и полнота работы системы

Отношение	**Ручная разметка, количество отношений**	**Система, количество отношений**	**Система, количество неправильных извлечений**	**Точность**	**Полнота**
almaMater	171	98	12	0.88	0.51
spouse	141	59	10	0.83	0.34

Анализ ошибок при работе системы (ошибочно извлеченных или, наоборот, пропущенных отношений) показал, что большая часть из них является ошибками работы модулей обработки текста – в частности, maltparser – около 30% всех ошибок. Кроме того, значительная часть ошибочно извлеченных отношений возникла из-за лексической многозначности. Сравним два предложения из тестового корпуса: «Фурукава *закончил* университет Нихон» - «Благодаря Божьему водительству, университет *закончил* оформление бумаг уже осенью». В первом случае имеет место отношение <Фурукава; almaMater; университет Нихон>. Во втором случае ошибочно извлекается отношение <университет; almaMater; оформление>. Подобных ошибок можно избежать, взвешивая извлеченные отношения в зависимости от количества их извлечений и количества шаблонов, с которыми они совпадают. Тогда до некоторого порога отношение считается гипотезой, а не достоверным фактом. Для тестирования подобной модификации требуется тестовый корпус достаточного объема, чтобы гарантировать неоднократное извлечение отношений для одной пары объект-субъект.

4 Описание практической части

Глава 4 рассматривает реализацию системы, описанной в главе 3, использованный инструментарий и характеристики функционирования.

4.1 Обоснование выбранного инструментария

Данный раздел описывает инструменты, выбранные для построения системы и проведения эксперимента.

Для поиска предикатов и получения всех отношений по заданному предикату использовалась точка доступа SPARQL, входящая в состав системы OpenLink Virtuoso[9] (свободно распространяемая версия Virtuoso Universal Server). Использовалась русскоязычная часть DBpedia Dataset 3.9[10], загруженная в базу данных mysql.

Для полнотекстового поиска изначально планировалось использовать снимок русской версии Википедии и расширение Cirrus для mediawiki-1.22.6, однако это значительно повысило требования к необходимым для работы системы ресурсам и выбор был сделан в пользу Bing API, что позволило к тому же не ограничивать область поиска статьями Википедии. Bing API был выбран среди остальных возможных поисковых систем в виду высокого лимита запросов – 5000 запросов для бесплатной учетной записи. Для взаимодействия с API использовали библиотеку azure-bing-search-java.jar[11].

Для токенизации использовали Tokenizer, предложенный в составе TreeTagger. Поскольку инструмент решает достаточно примитивную задачу и делает это успешно - искать более эффективную замену нецелесообразно.

Для определения частей речи был использован TreeTagger [26].

Для лемматизации использовали предложенный автором [24] Lemmatizer, работающий на базе CST Lemmatiser [25].

Для получения информации о семантических отношениях был использован maltparser[24]: он применим к большому входному корпусу. Кроме того, на текущий момент не известно других эффективных средств определения семантических отношений, применимых к рус-

[9] http://virtuoso.openlinksw.com
[10] http://wiki.dbpedia.org/Downloads39
[11] https://code.google.com/p/azure-bing-search-java/

скому языку. Для maltparser и TreeTagger использовались модели, тренированные на Национальном корпусе русского языка (НКРЯ)[12] предложенные автором работы [24].

Утилиты для построения модели и извлечения отношений были написаны на языке Java, который был выбран в силу кроссплатформенности и удобства совмещения с другими элементами системы, представленными в виде jar-библиотек. Модули получили названия RRE-Learner и RRE-Extractor, соответственно, где RRE – Russian Relation Extraction.

4.2 Структура системы

Данный раздел описывает структура модулей, реализованных в рамках дипломной работы для построения модели и извлечения отношений. Общая схема работы системы рассматриваться не будет, т.к. полностью соответствует представленной на рисунке 6 в главе 3.

[12] http://www.ruscorpora.ru

4.2.1 Структура модуля построения модели

На рисунке 8 представлена диаграмма классов, описывающая модуль построения модели.

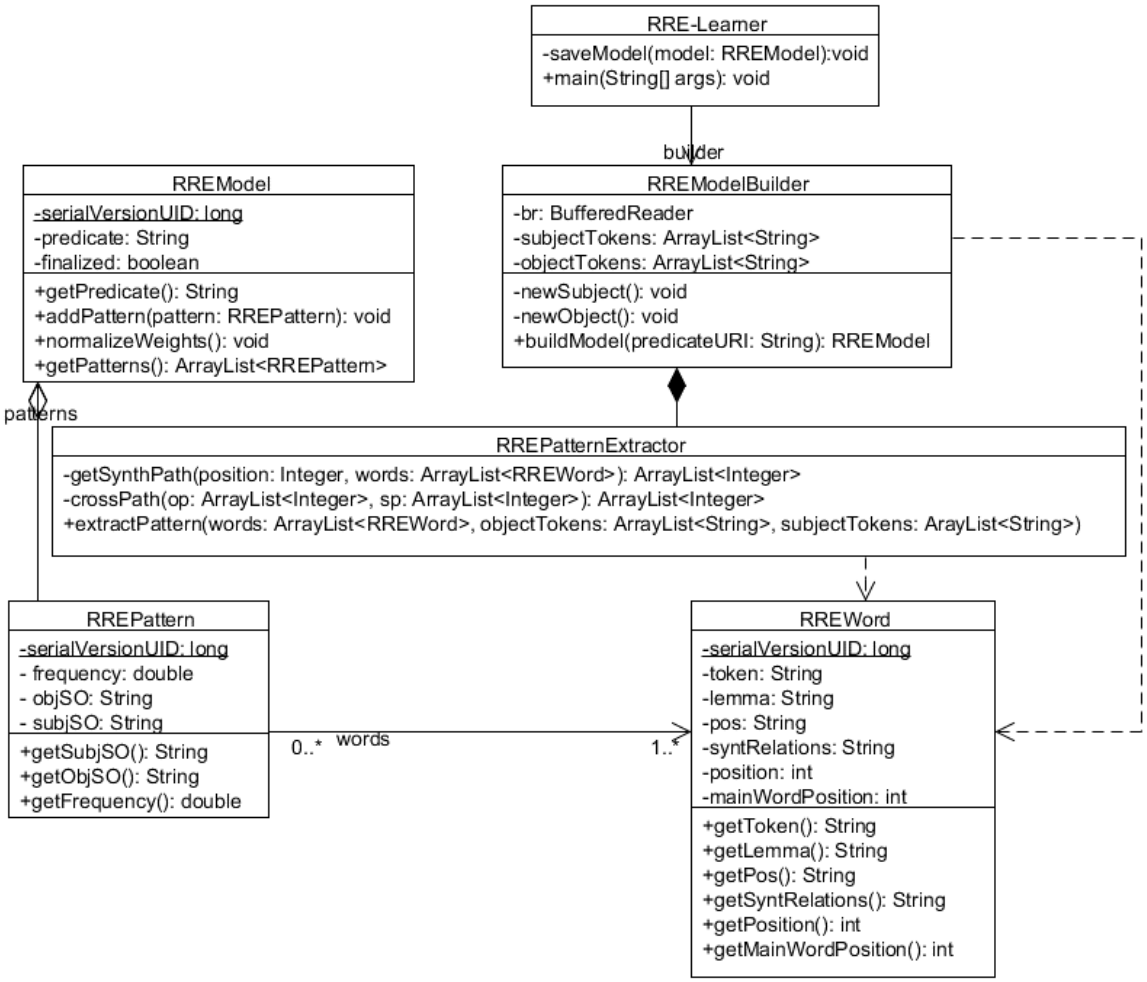

Рисунок 8: Диаграмма классов модуля RRE-Learner

Класс *RRE-Learner* координирует работу модуля. На вход методу main поступает путь к файлу, содержащему результат работы модуля обработки текста, и путь к файлу, в который требуется сохранить построенную модель, а также строка, соответствующая URI отношения, для которого строится модель.

Класс *RREModelBuilder* отвечает за построение модели. В нем происходит чтение входного текста и разбиение его на предложения, определение объекта и субъекта для каждого предложения, приведение предложений к массиву элементов *RREWord* для дальнейшего извлечения отношений.

RREPatternExtractor реализует извлечение отношения из предложения по заданным объекту и субъекту. Для каждого слова, обозначающего объект или субъект, строится кратчайший путь до корня синтаксического дерева при помощи метода *getSynthPath()*. После

чего пути объектов пересекаются с путями субъектов для выделения минимального связывающего их поддерева в методе *crossPath()*.

Класс *RREWord* служит для представления каждого слова предложения и грамматической информации о нем: леммы, части речи, синтаксического отношения, позиции в предложении и позиции слова, от которого зависит текущее.

Класс *RREPattern* описывает шаблон и хранит массив слов *RREWord*, синтаксические отношения для объекта и субъекта, а также вес, с которым шаблон входит в модель.

Класс *RREModel* представляет модель и содержит URI предиката и массив шаблонов. Кроме того, в модели имеется флаг finalized, позволяющий отслеживать состояние весов шаблонов: нормализацию весов можно провести только один раз и после этого добавлять новые шаблоны уже нельзя.

4.2.2 Структура модуля извлечения отношений

На рисунке 9 представлена диаграмма классов, описывающая модуль извлечения отношений из текста.

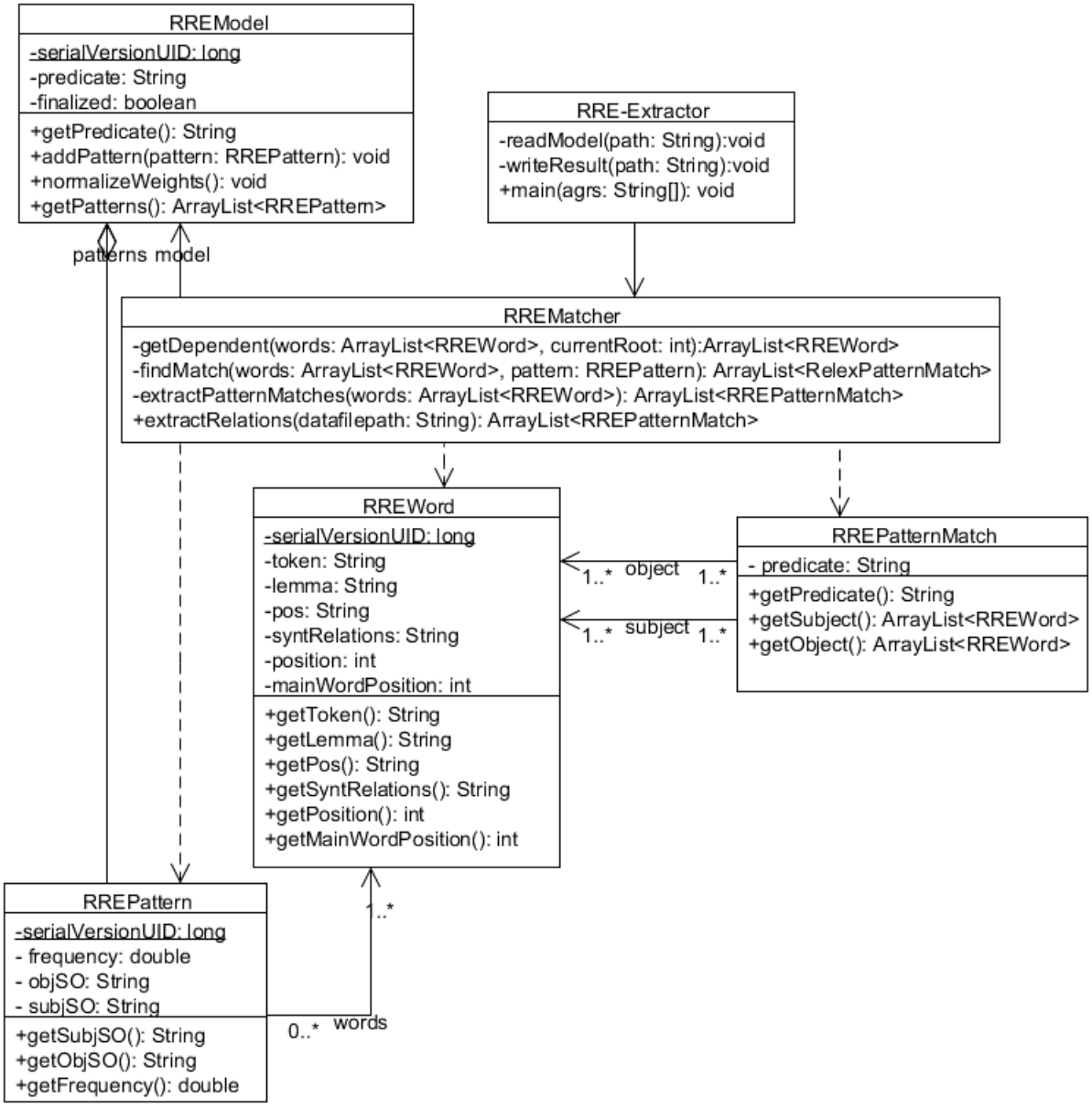

Рисунок 9: Диаграмма классов модуля RRE-Extractor

Класс *RRE-Extractor* координирует работу модуля, загружает модель и записывает результат извлечения в файл. На вход получает пути к файлам, содержащим модель и текст для извлечения.

Класс *RREMatcher* инициируется моделью, отвечает за поиск в тексте фрагментов, соответствующих шаблонам из модели.

Класс *RREPatternMatch* хранит результат извлечения отношения: предикат и слова, входящие в объект и субъект.

Остальные классы были рассмотрены в пункте 4.2.1.

4.3 Характеристики функционирования

Данная глава описывает такие характеристики функционирования построенной системы, как сложность и производительность.

4.3.1 Производительность

Тестирование системы производилось на тестовом тексте объемом 1500 предложений (всего 10 616 слов).

Итоговое время работы было рассчитано как среднее арифметическое результатов 10 тестовых запусков.

Конфигурация тестового компьютера: Intel Core i3 330M (4 CPU) 2130 Mhz, 4096Mb RAM.

В таблице ниже представлено среднее время работы для основных модулей системы.

Таблица 4: Время работы модулей системы

Модуль	**Среднее время работы (мс)**
Модуль обработки текста (токенизация, лемматизация, определение частей речи и синтаксических отношений, построение дерева зависимости)	374 964
Построение модели	229.4
Извлечение отношений из текста	326.6

Время выполнения запроса триплетов по предикату варьируется в зависимости от количества триплетов, удовлетворяющих параметрам запроса, в онтологии и занимает от 129 мс до 3 с.

Производительность модуля поиска предложений зависит от параметров доступа в Интернет и занимает в среднем 8 секунд.

4.3.2 Сложность

Модуль обработки текста состоит из токенизации, лемматизации, определения частей речи, построения дерева зависимости и определения синтаксических отношений. Каждый из использованных для этого инструментов работает за линейное время от количества слов.

Алгоритм построения модели в худшем случае имеет сложность $O(nk^3)$, где n – количество предложений, k – максимальное количество слов в предложении.

Алгоритм извлечения отношений в худшем случае проходит по предложению столько раз, сколько шаблонов в модели, поэтому для n предложений и p шаблонов имеет сложность $O(pn)$.

В итоге сложность работы системы в целом составляет $O(n+nk^3+pn)$.

Заключение

В рамках дипломной работы были получены следующие результаты:

1) Исследованы существующие методы извлечения отношений из текстов на естественных языках.
2) Разработан алгоритм извлечения отношений из текстов на русском языке на основе DBpedia.
3) Построена система извлечения отношений из текстов на основе DBpedia, подтверждающая работоспособность разработанного метода. Проведены экспериментальные исследования построенной системы.

Разработанный метод может быть использован и для других онтологий, хранящих отношения в виде триплетов.

Список цитируемой литературы

1. Berners-Lee T. et al. The semantic web //Scientific american. – 2001. – Т. 284. – №. 5. – С. 28-37.
2. Roussopoulos N. D. A semantic network model of data bases. – 1977
3. Gruber T. R. A translation approach to portable ontology specifications //Knowledge acquisition. – 1993. – Т. 5. – №. 2. – С. 199-220.
4. Lehmann J. et al. Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia //Semantic Web Journal. – 2013.
5. Agichtein E., Gravano L. Snowball: Extracting relations from large plain-text collections //Proceedings of the fifth ACM conference on Digital libraries. – ACM, 2000. – С. 85-94.
6. Brin S. Extracting patterns and relations from the world wide web //The World Wide Web and Databases. – Springer Berlin Heidelberg, 1999. – С. 172-183.
7. Carlson A. et al. Toward an Architecture for Never-Ending Language Learning //AAAI. – 2010.
8. Carlson A. et al. Coupled semi-supervised learning for information extraction //Proceedings of the third ACM international conference on Web search and data mining. – ACM, 2010. – С. 101-110.
9. Callan J. et al. Clueweb09 data set. – 2009.
10. Quinlan J. R., Cameron-Jones R. M. FOIL: A midterm report //Machine Learning: ECML-93. – Springer Berlin Heidelberg, 1993. – С. 1-20.
11. Horn A. On sentences which are true of direct unions of algebras //Journal of symbolic logic. – 1951. – С. 14-21.
12. Schmitz M. et al. Open language learning for information extraction //Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. – Association for Computational Linguistics, 2012. – С. 523-534.
13. Fader A., Soderland S., Etzioni O. Identifying relations for open information extraction //Proceedings of the Conference on Empirical Methods in Natural Language Processing. – Association for Computational Linguistics, 2011. – С. 1535-1545.

14. Suchanek F. M., Sozio M., Weikum G. SOFIE: a self-organizing framework for information extraction //Proceedings of the 18th international conference on World wide web. – ACM, 2009. – С. 631-640.

15. Suchanek F. M., Kasneci G., Weikum G. Yago: a core of semantic knowledge //Proceedings of the 16th international conference on World Wide Web. – ACM, 2007. – С. 697-706.

16. Exner P., Nugues P. Entity extraction: From unstructured text to DBpedia RDF triples //The Web of Linked Entities Workshop (WoLE 2012). – 2012.

17. Augenstein I., Padó S., Rudolph S. Lodifier: Generating linked data from unstructured text //The Semantic Web: Research and Applications. – Springer Berlin Heidelberg, 2012. – С. 210-224.

18. Cattoni R. et al. The KnowledgeStore: an Entity-Based Storage System //LREC. – 2012. – С. 3639-3646.

19. Fillmore C. J. Frame semantics and the nature of language //Annals of the New York Academy of Sciences. – 1976. – Т. 280. – №. 1. – С. 20-32.

20. Palmer M., Gildea D., Kingsbury P. The proposition bank: An annotated corpus of semantic roles //Computational Linguistics. – 2005. – Т. 31. – №. 1. – С. 71-106.

21. Björkelund A., Hafdell L., Nugues P. Multilingual semantic role labeling //Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task. – Association for Computational Linguistics, 2009. – С. 43-48.

22. Lee H. et al. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task //Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task. – Association for Computational Linguistics, 2011. – С. 28-34.

23. Ratinov L. et al. Local and global algorithms for disambiguation to wikipedia //Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. – Association for Computational Linguistics, 2011. – С. 1375-1384.

24. Sharoff S., Nivre J. The proper place of men and machines in language technology. Processing Russian without any linguistic knowledge //Komputernaja lingvistika i intel-

lektual’nye tekhnologii: Po materialam Mezhdunarodnoj konferencii “Dialog”(Bekasovo, 25-29 maja 2011). – 2011. – С. 591-604.

25. Jongejan B., Haltrup D. the CST Lemmatiser //Center for Sprogteknologi, University of Copenhagen version. – 2005. – Т. 2.
26. Schmid H. Probabilistic part-of-speech tagging using decision trees //Proceedings of international conference on new methods in language processing. – 1994. – Т. 12. – С. 44-49.