МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)
Факультет информационных технологий
Кафедра систем информатики
(название кафедры)

Направление подготовки 230100.62 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Разработка редактора правил для извлечения информации из текстов на естественном языке.
(тема работы)

Ли Дмитрий Владимирович
(фамилия, имя, отчество автора - студента –выпускника)

«К защите допущен»
Лаврентьев М. М., д.ф.-м.н.,
профессор, зав. кафедрой
……/……..
(фамилия , И., О.) / (подпись, МП)
«……»……20…г.

Научный руководитель
Сидорова Е. А., к.ф.-м.н.
с.н.с. ИСИ СО РАН
……/……..
(фамилия , И., О.) / (подпись, МП)
«……»……20…г.

Дата защиты: «……»……20…г.
Автор……/……..
(фамилия , И., О.) / (подпись)

Новосибирск, 2014

Оглавление

Введение

Извлечение информации из текстов определённой предметной области играет значительную роль во многих прикладных задачах. Одним из способов организации данного процесса является составление правил извлечения информации в виде фактов и их дальнейшее автоматическое применение к текстам. Правила извлечения фактов создаются экспертом. Формирование правил в большинстве случаев требует больших трудозатрат, кроме того, большое количество правил, созданных разными экспертами, зачастую приводит к появлению правил противоречащих друг другу. В связи с этим, разработка редактора для составления правил извлечения информации, который позволил бы упростить и автоматизировать работу эксперта является актуальной задачей.

Цель работы: Создание редактора правил для извлечения из текста информации.

Данная работа является частью большего проекта по разработке интеллектуальной системы, осуществляющей анализ текста и извлечение информации на основе правил. На текущий момент, для успешного составления правил помимо эксперта, требуется инженер знаний, который переводит правила в понятный системе формат. Поэтому создание редактора в разы ускорит взаимодействие эксперта и системы.

Для достижения поставленной цели было необходимо выполнить следующие задачи:

- Исследовать предметную область, провести обзор существующих систем извлечения информации из текстов на основе правил.
- Разработать формат представления онтологий, семантических и морфологических признаков словаря в виде XML-документа, для последующей их работы с редактором.
- Проектировать и реализовать пользовательский интерфейс редактора
- Обеспечить корректность составленных фактов на основе онтологий и признаковой системы словаря.
- Разработать алгоритм поиска циклических зависимостей между правилами извлечения.

1. Обзор существующих систем извлечения информации из текстов

Задача извлечения информации из текста сама по себе не нова: в этом направлении проделано довольно много работы как со стороны крупных компаний Яndex и Google, так и со стороны независимых разработчиков. На данный момент существует множество различных систем извлечения фактов, все они обладают разным функционалом и по-своему отличаются.

Для исследования были выбраны два известных решений, а именно: RCO Fact Extractor и лингвистический процессор Semantix.

Процессор Semantix распространяются бесплатно. Стоимость лицензии RCO Fact Extractor Desktop на одно рабочее место по состоянию на 2014 год составляет от 70000р.

Перед началом подробного обзора выбранных систем необходимо отметить их некоторые особенности. Компания "ЭР СИ О" помимо собственно программы с графическим интерфейсом для Windows, выпускает динамическую библиотеку для разработчиков (SDK). В то время как, лингвистический процессор Semantix представляет собой библиотеку COM-объектов и функций.

1.1. RCO Fact Extractor

Начнем рассмотрение с самого известного решения в данной области RCO Fact Extractor. Fact Extractor Desktop – это персональное приложение для Windows, которое предназначено для аналитической обработки текста на русском языке и выявления фактов различного типа, связанных с заданными объектами – персонами и организациями. Основная сфера применения программы – это задачи из области компьютерной разведки, требующие высокоточного поиска информации, например, автоматический подбор материала к досье на целевой объект или же мониторинг определенных сторон его активности, освещаемых в СМИ. Программа позволяет не только найти фрагменты текста, в которых говорилось, например, о поездках персоны, ее встречах, заключении договоров, сделках купли-продажи, но и точно определить все места поездок, визави и контрагентов, наименование товаров и прочее.

Программа работает в среде Windows 2000 и выше и позволяет обрабатывать документы в популярных текстовых форматах из различных источников – файловой системы, заданных web-сайтов, базы данных.

Результат работы программы – таблица, которая содержит информацию о найденных фактах, связанных с объектами мониторинга, и может экспортироваться в html-формат для формирования отчета или для загрузки в стороннее приложение, работающее с уже структурированными данными.

Данная программа предполагает настройку шаблонов для поиска и классификации фактов самых различных типов. Такие специализированные шаблоны либо приобретаются отдельно, либо создаются пользователем самостоятельно при помощи дополнительной программы RCO Fact Tuner. Тем не менее, стандартные шаблоны, включенные в комплект поставки Fact Extractor Desktop, позволяют распознавать огромное количество самых разнообразных фактов, но без детальной классификации, т.е., попросту находить события, в которых участвует целевой объект, и извлекать из текста всех прочих фигурантов этих событий, без детализации их ролей.

Помимо собственно программы с графическим интерфейсом для Windows, компания "ЭР СИ О" выпускает пакет для разработки программного обеспечения (SDK), на базе которого построен Fact Extractor и который позволяет включать возможности анализа текста в собственные приложения.

Ядро пакета представляет библиотека RCO FX Ru, которая осуществляет полный синтактико-семантический разбор русского текста. Библиотека выделяет различные классы сущностей, упомянутых в тексте (персоны, организации, география, предметы, действия, атрибуты и др.), и строит сеть отношений, связывающих эти сущности, а также предоставляет всю грамматическую информацию о составляющих текста. Средства библиотеки также обеспечивают семантическую интерпретацию результатов разбора текста – поиск описаний ситуаций, удовлетворяющих заданным семантическим шаблонам.

1.2. Semantix

Процессор Semantix предназначен для извлечения знаний из текстов на естественном языке (русском, английском) с автоматическим формированием Базы Знаний. Из текстов (документов) извлекаются интересующие пользователя объекты, их свойства и связи. Представляются факты участия объектов в действиях. В результате на основе каждого документа строится семантическая сеть, отражающая его семантическую структуру. Качество лингвистического процессора определяется рядом факторов. Во-первых, это возможности выделения объектов и связей. Имеется в виду типы выделяемых объектов, их количество. Процессор Semantix выделяет до 40 типов объектов, в том числе

комлексных объектов, соответствующих действиям и событиям. С увеличением количества возникают дополнительные трудности, связанные с "коллизией" правил выделения: одни правила могут захватывать слова, относящиеся к другим объектам и выделяемым другими правилами. становится важным порядок применения правил, в том числе, правил идентификации. Во-вторых, важный фактор - это избирательность правил и процедур идентификации: коэффициент шумов и потерь. Под шумами понимается наличие лишних слов в объектах. Потери - это когда объект не выявлен или выявлен частично: в тексте есть слова, которые не вошли в объект. В процессоре Semantix правила устроены таким образом, что они обеспечивают высокую степень избирательности и минимизацию шумов и потерь при большом количестве выделяемых объектов. Третий фактор - возможность анализа не только русского, но и английского языка. Четвертый фактор - скорость работы лингвистического процессора, т.е. время анализа.

Подводя итог краткому исследованию существующих решений можно выделить общий ряд преимуществ и недостатков. Все системы обладают достаточной функциональностью для областей, где требуется автоматическая обработка потоков текстов на естественном языке. Из недостатков выделяется работоспособность только на операционных системах Windows и ни на каких-либо других. Также из недостатков можно отметить, что все перечисленные системы работают только с текстами на русском языке.

2. Модель представления знаний

Лингвистическая база знаний содержит всю совокупность лингвистических знаний, необходимых для анализа текста, и включает онтологии, словари предметной лексики и схемы извлечения фактов.

2.1. Онтология

Онтология включает в себя *понятия и отношения* предметной области и описывает данные, которые необходимо извлечь из текста.

Онтология [1] - точное, подробное описание (модель) некоторой области знаний. Формально онтология - это пятерка вида: $O = \langle C, A, R_C, T, D \rangle$, где

- C – множество классов, описывающих понятия некоторой предметной или проблемной области;
- A – множество атрибутов, описывающих свойства понятий и отношений;
- $R_C = \{r_C \mid r_C \subseteq C \times C\}$ – множество отношений, заданных на классах (понятиях);
- T – множество стандартных типов значений атрибутов (*string, integer, real, date, bool*);
- D – множество доменов (множеств значений стандартного типа *string*);

Пример предметной онтологии "Организации":

class Персона (Фамилия: string; Имя: string; Отчество: string; Инициалы: string;
ПолноеИмя: string; Звание: set of domen_Звания)

class Регион (Название: string; Тип: domen_Географический_Тип)

relation Включение-Регион < Регион, Регион >

class Организация (Название: string; Аббревиатура: string)

class Институт : Организация
class Филиал : Организация
class Библиотека : Организация
…

relation Включение-Организация < Организация, Организация >

relation Сотрудник <Персона, Организация>
(Должность: domen_Должности; Дата1: data; Дата2: data)

relation Локализация <Организация, Регион > (Дата1: data; Дата2: data)

class Документ (Текст: string; Язык: domen_Языки)

relation Отражен < Entity, Документ >

2.2. Словарь предметной области

Тезаурус (словарь предметной области) является лексической онтологией предметной области и описывает терминологию, применяемую специалистами в данной области. Элементами тезауруса являются термины и их лингвистически-обусловленные взаимосвязи.

Формально, словарь-тезаурус – это знаковая система

T = <V, S, M, Gsyn>, где

V– конечное множество терминов,

S – конечное множество семантических классов словаря (семантические характеристики позволяют определить элементы онтологии, описываемые с помощью данного термина),

M - конечное множество морфологических признаков словаря,

Gsyn={G_1,…,G_k} – конечное множество подмножеств терминов, связанных отношением синонимии (синсеты). Синсет позволяет сопоставить одному смысловому значению предметной области набор альтернативных терминов, которые могут служить для репрезентации данного значения в тексте.

2.3. Модель извлечения фактов

Факт представляет собой знание об объектах, их свойствах и ситуациях, зафиксированное в высказывании (языковом выражении). Формализация понятия факта позволяет связать его с понятиями и отношениями, заданными в онтологии. Каждый факт имеет свой тип – название отношения и список его аргументов. Таким образом, понятия информация и факт тесно связаны.

Модель извлечения факта из текста должна включать в себя множество языковых способов репрезентации данного отношения носителями подъязыка и способы (правила) трансформации их в схему факта. Такую модель мы будем называть схемой извлечения факта(СИФ)

Формально, СИФ – это тройка вида < A, Res, C > [2, 3], где

A = {c_1, …, c_n}, где c_i = <t, s, h>, где t – задает тип элемента (термин/объект), s – семантический класс сущности, а h – параметр регулирующий использование иерархии наследования, определяемой для классов сущностей.

Res = < t, op(t), P > – результат применения СИФ, где

- t – задает тип элемента (класс нового объекта или один из аргументов);
- op(t) – тип операции (создание и/или редактирование аргумента), применяемой, если выполнены ограничения C;
- P – множество правил для формирования/редактирования объекта. Каждое правило ставит в соответствие атрибуту результирующего объекта либо точное значение, либо значение атрибута одного из аргументов.

C – множество ограничений, накладываемых на характеристики аргументов факта.

Выделяются следующие ограничения:

- условия на морфологические и семантические характеристики аргументов схемы (например, arg1.Падеж = рд, arg1.SemClass=Лок)
- ограничение синтаксической сочетаемости вершин синтаксических групп, реализующих аргументы схемы (например, Synt = Согл(число, падеж)
- структурно-текстовые ограничения на взаиморасположение аргументов в тексте: позиция аргументов относительно друг друга, тип контактности (например, Contact=Common – аргументы могут разделяться в тексте знаками препинания и/или незначимыми словами), тип сегмента.

Пример схем:

Scheme конференция

arg1: Term*::Номер_конференции*()

arg2: Object*::Конференция*()

Condition Position = preposition_priority, Contact = absolute

⇨ Arg2::*Конференция*(Номер: arg1.Номер)

Вторая схема имеет два аргумента, которые описывают элементы из словаря предметной лексики и онтологии. Позиционное ограничение определяет контактность терминов в тексте, а также указывает на тот факт, что расположение номера конференции слева от события (конференция) является приоритетным (IV Международная научно практическая конференция).

В результате второй схемы будут объекты класса Конференция будут дополняться атрибутом "Номер конференции". Эта схема применима к любому контексту, в котором конференция сопровождается номером.

3. Редактор правил

Редактор представляет собой кроссплатформенное графическое приложение, позволяющее создавать правила для извлечения информации из текста.

В редакторе представлена возможность открывать сохраненные схемы и редактировать их. Схемы автоматически проверяются на циклическую зависимость.

В состав программы входит следующее:

- пользовательский интерфейс,
- модуль синтаксического разбора XML-документов,
- модуль генерации XML-документов.

Модуль синтаксического анализа XML-документов отвечает за чтение и обработку онтологий и готовых схем фактов.

Модуль генерации XML-документов создает XML-файл на основе выбранных экспертом параметров.

Пользовательский редактор построен на четырех основных типах интерфейсных компонентов: основное окно для вывода информации, диалог открытия\сохранения файлов (архив схем, онтологий морфологических таблиц, диалог ввода новых данных, диалог с выбором из списка\дерева.

3.1. Требование к программе

Функционирование программы не должно приводить к сбою (фатальному нарушению работы системы). Организация диалога должна предусматривать защиту от ввода некорректных данных. Программа должна выдавать диагностику состояния системы и сообщения о любых возникших ошибках.

3.2. Средства разработки

Программа написана на языке C++ с использованием кроссплатформенного инструментария разработки программного обеспечения - Qt. Выбор данного языка объясняется тем, что язык C++, в сравнении с другими языками программирования более быстрый.

3.3. *Реализация основной функциональности*

Главное окно программы показано на рисунке 1. В основном окне, внутри вкладок отображаются загруженные и\или созданные схемы извлечения фактов, онтология, словарь, которые представлены в древовидной структуре.

Отображенные схемы можно редактировать воспользовавшись контекстным меню. А также можно добавлять новые схемы в текущий банк схем с помощью кнопки *Add new scheme.*

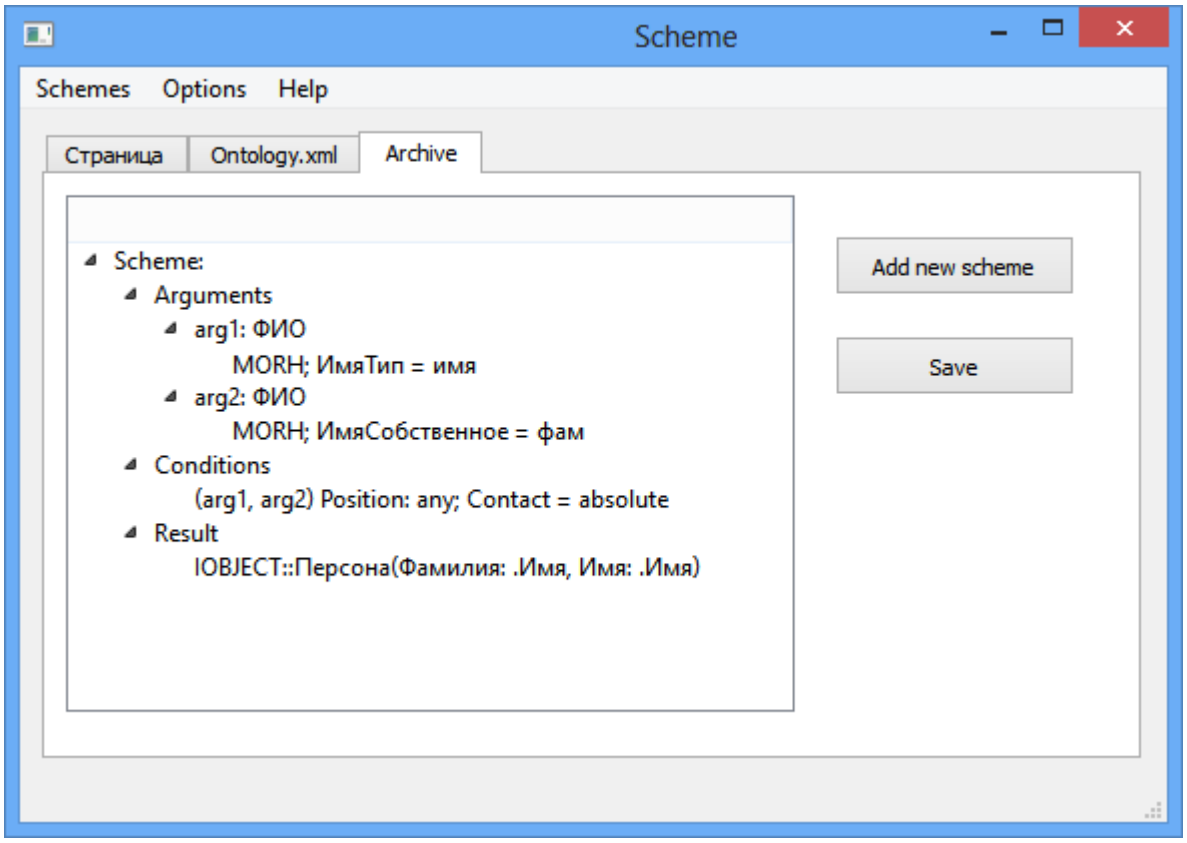

Рисунок 1. Пример основного окна программы

Главное меню

Главное меню располагается вверху программы.

Главное меню содержит разделы:

- Schemes - создание и загрузка схем.
- Options - загрузка онтологии, семантических/морфологических признаков словаря.
- Справка - информация о программе.

Раздел Schemes

Возможные действия со схемами через раздел "Schemes":

- Создать новый банк схем
- Открыть существующий банк схем

Добавление аргумента

При создании банка схем предлагается ввести имя банка, затем открывается диалоговое окно для уточнения параметров схемы (рис. 2). В диалоговом окне отображаются поле для ввода названия, а также вкладки для создания\редактирования аргументов, структурных ограничений и результатов схемы.

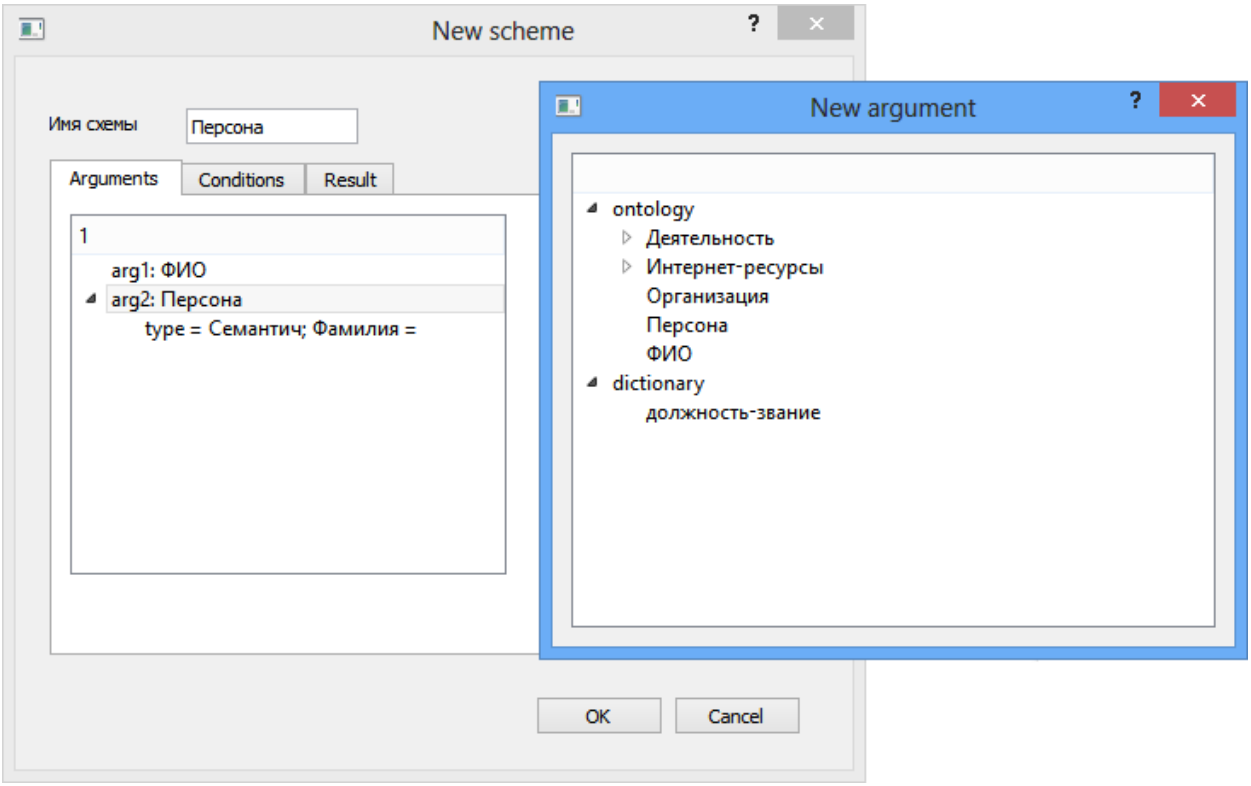

Рисунок 2. Пример окна для создания схемы.

Во вкладке *Arguments* при нажатии на кнопку *Add new* появляется окно, в котором необходимо выбрать класс аргумента из загруженной ранее онтологии. После выбора класса, с помощью контекстного меню можно добавить условия на аргумент (рис. 3).

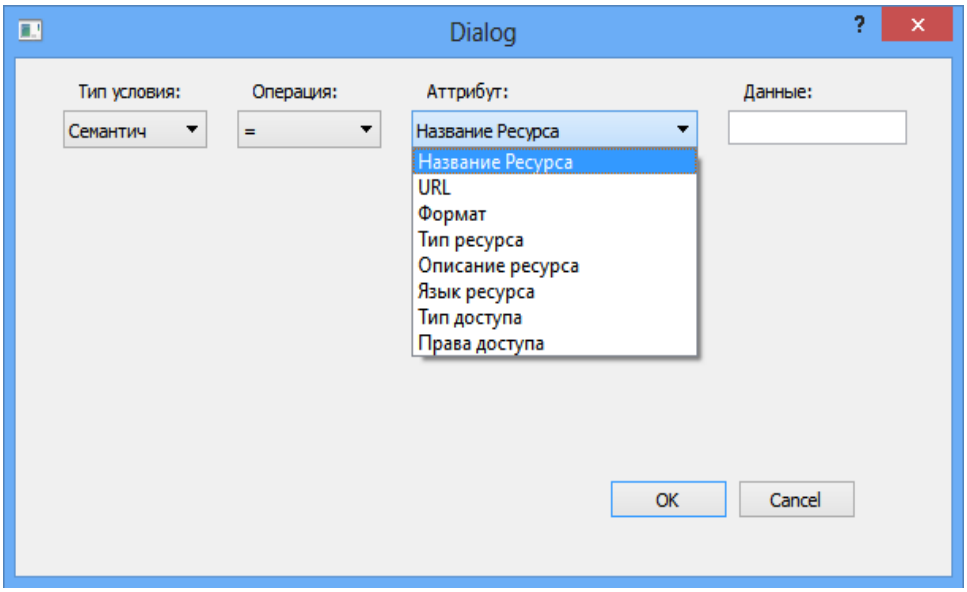

Рисунок 3. Добавление условия на аргумент.

В открывшемся окне тип операции меняется в зависимости от типа атрибута, указанного в онтологии (String, Int, Bool, Domen). Список атрибутов формируется на основе выбранного аргумента.

Добавление условий на аргумент

Условия бывают четырех типов: семантические (онтологические), семантико-синтаксические, морфологические и структурные. Для того, чтобы добавить морфологические условия на аргумент, необходимо предварительно загрузить морфологическую таблицу в специальном формате. Пример файла морфологической таблице представлен в приложении В. Также можно удалять и редактировать созданные аргументы.

Добавление структурных ограничений

Вкладка *Conditions* предназначена для создания структурных ограничений на схему. При добавлении нового ограничения появляется окно для определения контактности

между аргументами и их позиции относительно друг друга (рисунок 4).

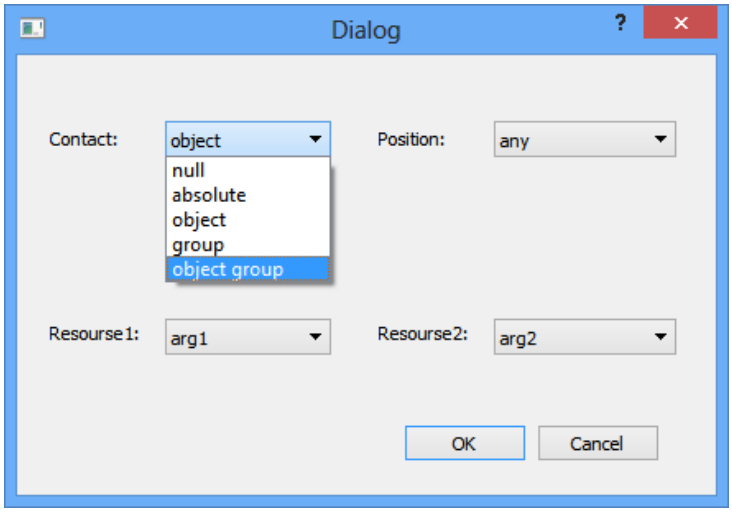

Рисунок 4. Добавление структурных ограничений.

Формирование результирующего объекта

Во вкладке *Result* (рисунок 5) с помощью кнопки *Set object* необходимо выбрать тип операции: создание нового объекта, изменение существующего. При создании объекта необходимо определить к какому классу принадлежит объект, выбрав его из списка онтологических классов. При редактировании - выбрать аргумент, который необходимо изменить. При нажатии на кнопку *Add rule* появляется окно, в котором можно создать атрибуты для результирующего объекта.

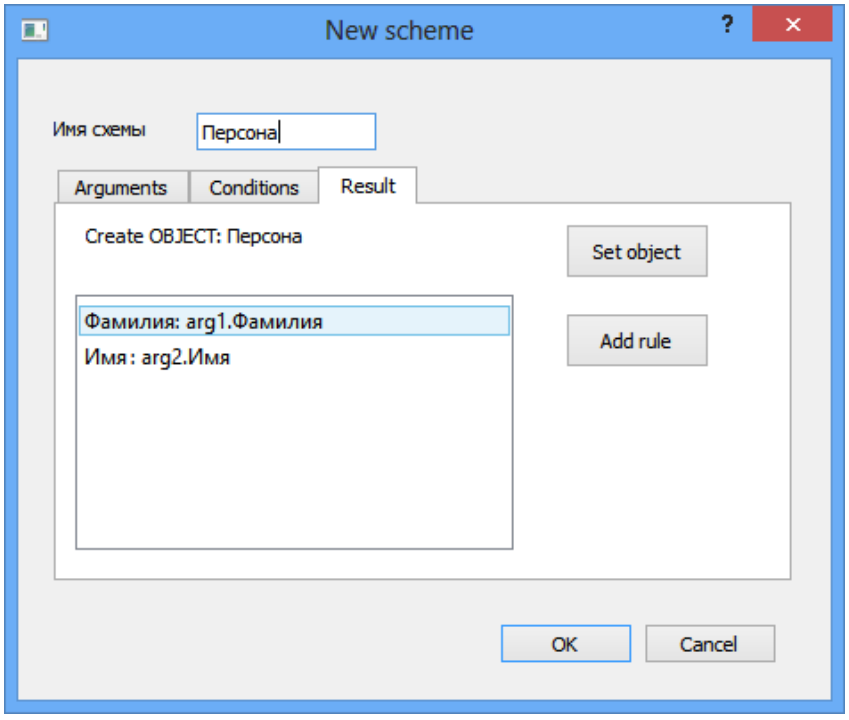

Рисунок 5. Формирование объекта

После подтверждения создания схемы в главном окне, в новой вкладке отобразится созданная схема. Также можно сохранить все полученные схемы в XML-файл.

Пример файла банка схем представлено в приложении А.

Настройки

Необходимые настройки для создания схем:

- Загрузка онтологии.
- Загрузка словаря предметной области.
- Загрузка морфологической таблицы.

При загрузке онтологии необходимо выбрать файл формата XML. Пример файла онтологии представлен в приложении Б. После успешной загрузки, онтология отображается в главном окне в новой вкладке (рис 6.). Аналогично с загрузкой словаря и морфологической таблицей.

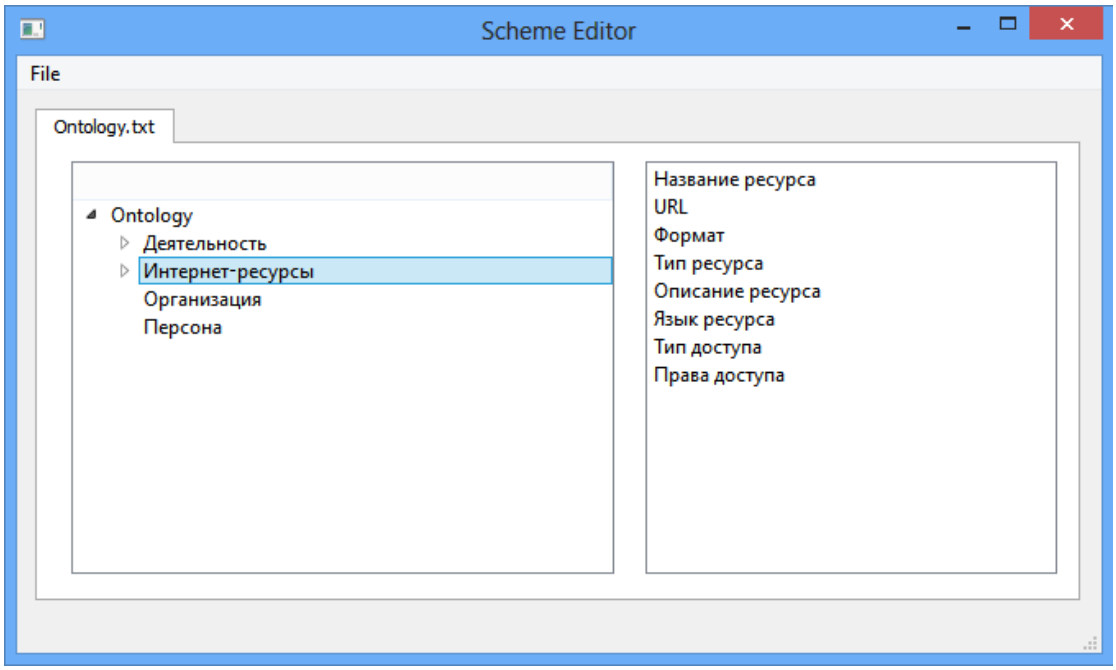

Рисунок 6. Пример загруженной онтологии

XML-формат описания морфологической таблицы представлено в приложении В.

3.4. Описание входных\выходных данных.

ВХОДНЫЕ ДАННЫЕ. Входными данными для программы являются:

- XML-файл описывающий онтологию
- XML-файл описывающий морфологическую таблицу и семантические классы словаря.
- XML-файл описывающий схему факта.

ВЫХОДНЫЕ ДАННЫЕ. Выходными данными являются:

- выводимый на экран банк схем фактов в виде дерева.
- XML-файл - схема извлечения факта, созданная на основе загруженной онтологии и выбранных настроек.

4. Эксперимент

В качестве эксперимента была рассмотрена задача извлечения информации о конференциях из различных документов. Онтология предметной области будет содержать следующие классы:

class Конференция(Место; Организатор; Название; Номер_конференции)

class Место_проведение(Город; учреждение)

class Организация(Название)

Пример схем извлечения информации о конференциях:

Scheme конференция

arg1: Term*::событие*()

⇨ Object::*Конференция*(Название: arg1.Название)

Scheme Место_проведения

arg1: Term*::город*()

⇨ Object::*Место_проведения*(Город: arg1.Город)

Scheme

arg1: Object*::Место_проведения*()

arg2: Object*::Конференция*()

Condition Position = any, Contact = object

⇨ Arg2::*Конференция*(Место: arg1.город; Организатор: arg1.Учреждение)

В результате будут формироваться объекты класса Конференция с определенными атрибутами: место, организатор.

Заключение

В ходе данной квалификационной работы были выполнены следующие задачи:

- изучена предметная область, проведен обзор существующих систем извлечения информации;
- разработан формат представления онтологий, семантических и морфологических признаков словаря в виде XML-документа, для последующей их работы с редактором;
- улучшен формат представления правил;
- спроектирован и реализован пользовательский интерфейс редактора;
- обеспечена корректность составленных фактов на основе онтологий и признаковой системы словаря.

Анализ существующих систем извлечения информации показал, что, хотя в данном направлении существует множество решений данной проблемы, на данный момент многие из них не являются кроссплатформенными и не работают с документами на различных языках.

Практическим результатом данной работы является программный интерфейс для создания схем извлечения информации из текстов. Данная программа обеспечивает корректность созданных схем, тем самым облегчая работу эксперту.

Дальнейшая работа может вестись в следующих направлениях:

- добавление дополнительных функций;
- улучшение графического интерфейса
- разработка методов обеспечения корректности схем

Список литературы

[1] Боровикова О.И. Организация порталов знаний на основе онтологий / Боровикова О.И., Загорулько Ю.А. // Компьютерная лингвистика и интеллектуальные технологии: Труды международного семинара «Диалог 2002» (Протвино, 6-11 июня 2002 г.). – Москва: Наука, – 2002. –Т.2, –С.76–82

[2] Кононенко И.С. Подход к извлечению фактов из текста на основе онтологии / Кононенко И.С., Сидорова Е.А. // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Международной конференции «Диалог 2009» (Бекасово, 27-31 мая 2009 г.). Вып. 8 (15). М.: РГГУ, 2009. –С. 451-457.

[3] Сидорова Е.А. Фактографический анализ текста в контексте интеллектуальных информационных систем / Сидорова Е.А. // Тр. XVIII Байкальской Всероссийской конференции "Информационные и математические технологии в науке и управлении". – Иркутск: Институт систем энергетики им Л.А. Мелентьева СО РАН. –2013. –Т.3. –С79-85.

[4] Кузнецов И.П. Особенности извлечения знаний из текстов семантико-ориентированным лингвистическим процессором Semantix / Кузнецов И. П., Ефимов Д. А. // Сб. Компьютерная лингвистика и интеллектуальные технологии. Выпуск 7 (14). По материалам конференции «Диалог 008»..РГГУ, М.:2008., С. 281-291.

Приложение А

Пример файла банка схем извлечения факта.

```
<FATON version="1.0" description="FATON Scheme of Facts">
    <Bank Name="MagArhive" Type="Schemes">

        <Scheme NumArg="1" CountType="0" Segment="">
            <Argument ObjectType="TERMIN_ALEX" ClassName="организация" TypeCompare="PLUS_CHILD"/>

            <Result Type="CREATE" ObjectType="IOBJECT" ClassName="Организация" >
                <Rule AttrName="Имя" AttrType="string" Resurce="Arg1" FromAttrName="Имя"/>
                <Rule AttrName="Аббревиатура" AttrType="string" Resurce="Arg1" FromAttrName="Name"/>
            </Result>

        </Scheme>

        <Scheme Name="asd" NumArg="2" CountType="0" Segment="">
            <Argument ObjectType="TERMIN" ClassName="фио" TypeCompare="EQUAL">
                <Condition Type="MORH" Operation="=" AttrName="фио-тип" AttrType="string" Data="фам" />
            </Argument>
            <Argument ObjectType="TERMIN_ALEX" ClassName="инициалы" TypeCompare="EQUAL"/>
            <ConditionStruct CondType="Position" Contact="CONTACT_ABSOLUTE" TextPos="POSITION_ANY" />
            <Result Type="CREATE" ObjectType="IOBJECT" ClassName="Персона" >
                <Rule AttrName="Инициалы" AttrType="string" Resurce="Arg2" FromAttrName="Value"/>
                <Rule AttrName="фамилия"  AttrType="string" Resurce="Arg1" FromAttrName="Name" />
            </Result>
        </Scheme>
    </Bank>

</FATON>
```

Приложение Б

Пример фрагмента файла онтологии ontology.xml.

```
<?xml version="1.0" encoding="utf-8"?>

<ontology>
  <class name="Деятельность">
        <attr type="string">Название деятельности</attr>
        <attr type="string">Описание деятельности</attr>
        <attr type="string">Дата начала</attr>
        <attr type="string">Стадия проекта</attr>
  </class>
  <class name="Интернет-ресурсы">
        <attr type="string">Название Ресурса</attr>
        <attr type="string">URL</attr>
        <attr type="string">Формат</attr>
        <attr type="string">Тип ресурса</attr>
        <attr type="string">Описание ресурса</attr>
        <attr type="string">Язык ресурса</attr>
        <attr type="string">Тип доступа</attr>
        <attr type="string">Права доступа</attr>
  </class>

  ...

  <class name="Организация">
        <attr type="string">Название</attr>
        <attr type="string">Аббревиатура</attr>
  </class>
  <class name="Проекты" base="Деятельность"/>
  <class name="Информационные Ресурсы" base="Интернет-ресурсы"/>
  <class name="Сайты организаций, персон, проектов" base="Интернет-ресурсы"/>
</ontology>
```

Приложение В

Пример файла морфологической таблицы gramtab.xml.

```
<?xml version="1.0" encoding="utf-8"?>

<Gramtab>
  <Attr name="Род">
        <Value>жр</Value>
        <Value>мр</Value>
        <Value>мр-жр</Value>
        <Value>ср</Value>
  </Attr>
  <Attr name="Число">
        <Value>ед</Value>
        <Value>мн</Value>
  </Attr>

...


  <Attr name="ИмяСобственное">
        <Value>имя_тип</Value>
        <Value>лок</Value>
        <Value>фам</Value>
        <Value>орг</Value>
  </Attr>

  <Attr name="ИмяТип">
        <Value>имя</Value>
        <Value>отч</Value>
  </Attr>
</Gramtab>
```