МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий
Кафедра Общей информатики

Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль): Программная инженерия и компьютерные науки

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Усовой Дарьи Сергеевны

Тема работы:

РАЗРАБОТКА МЕТОДОВ ИЗВЛЕЧЕНИЯ И ОБРАБОТКИ АРГУМЕНТОВ ИЗ ТЕКСТОВ ЕСТЕСТВЕННОГО ЯЗЫКА, ОСНОВАННЫХ НА ЛОГИКЕ ПРЕДИКАТОВ

«К защите допущена»
Заведующий кафедрой Общей информатики, д.ф.-м.н., доцент
Пальчунов Д.Е. / ……..
(ФИО) / (подпись)
«31» мая 2021 г.

Руководитель ВКР
Заведующий кафедрой Общей информатики, д.ф.-м.н., доцент
Пальчунов Д.Е. / ……..
(ФИО) / (подпись)
«31» мая 2021 г.

Соруководитель ВКР
Ассистент кафедры Общей информатики
Погодин Р.С./ ……..
(ФИО) / (подпись)

Новосибирск, 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий

Кафедра общей информатики

Направление подготовки 09.03.01 Информатика и вычислительная техника

Направленность (профиль): Программная инженерия и компьютерные науки

УТВЕРЖДАЮ
Зав. кафедрой Пальчунов Д. Е.
……
(подпись)
“10” ноября 2020 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

Студентке ……..…Усовой Дарье Сергеевне…….., группы 17206

Тема Разработка методов извлечения и обработки аргументов из текстов естественного языка, основанных на логике предикатов

Утверждена распоряжением проректора по учебной работе от 10.11.2020 № 319

Срок сдачи студентом готовой работы «31» мая 2021 г.

Исходные данные (или цель работы) Построение теории аргументации, основанной на формальном представлении знаний, реализация разработанных методов в программной системе, предназначенной для извлечения, анализа и поиска аргументации из текстов естественного языка.

Структурные части работы Титульный лист, содержание, введение, основная часть, заключение, список литературы, приложения

Консультанты по разделам ВКР (при необходимости, с указанием разделов)

……
…….

Руководитель ВКР
Заведующий кафедрой,
Д.ф.-м.н., доцент
Пальчунов Д.Е./…….
(ФИО) / (подпись)

«10» ноября 2020 г.

Задание принял к исполнению
……Усова Д.С./…….
(ФИО) / (подпись)
«10» ноября 2020 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Тематика моделирования рассуждений на данный момент актуальна, для нее не существует универсального решения. Тем не менее, возможность предоставлять автоматическую аргументацию была бы полезна в таких областях как медицина, политика, социология и т.д. Многие существующие решения предлагают лишь анализ синтаксической структуры предложений, при этом теряется семантика утверждений. Многие подходы оперируют числами, но при этом не учитывают множество деталей, которые важны для понимания смысла аргументации. Также, существует ряд проблем, например проблема омонимичных слов, решение которых не является тривиальным.

Существует несколько популярных подходов к решению задач моделирования рассуждений. Среди них можно выделить два направления - нейронные сети и методы классической логики. Нейронные сети, как правило, представляют собой черный ящик, объяснить причины того или иного ответа не представляется возможным. Классическая логика не позволяет выводить новых знаний, она лишь способна получить другую формулировку уже имеющихся знаний. Подходы классических систем на основе прецедентов на данный момент устарели.

Объектом исследования являются тексты на естественном языке и знания, содержащиеся в них. В качестве таких текстов рассматриваются новостные статьи.

Предметом исследования является извлечение, обработка и поиск аргументации.

Целью данной работы является разработка методов аргументации, основанных на формальном представлении знаний, их реализация в программной системе, предназначенной для извлечения, анализа и поиска аргументации из текстов естественного языка.

В рамках работы над темой были поставлены следующие задачи:

1. Изучение существующих решений для аргументации рассуждений;
2. Изучение основных понятий в области аргументации рассуждений;
3. Разработка алгоритмов для извлечения, обработки и поиска аргументов;
4. Построение теории аргументации, формализующей разработанные подходы;
5. Разработка программной системы на основе описанных подходов;
6. Тестирование системы.

Основные проблемы, касающиеся темы аргументации рассуждений:

1. Аргументация может проходить в условиях наличия неполных и противоречивых данных;
2. Проблема омонимичных слов;

3. Существование синонимичных и похожих слов для понятий в языке;
4. Проблема наличия контекста в текстах естественного языка;
5. Проблема получения новых знаний.

В процессе обзора существующих решений был рассмотрен ряд работ и публикаций на тему моделирования рассуждений в текстах естественного языка.

Одним из наиболее популярных подходов является применение классической логики для работы с аргументацией. В этом направлении можно отметить работы Э.Хантера [1]. Основная суть предложенного им алгоритма выглядит следующим образом: в начале работы имеется база аргументов в виде множества выражений логики предикатов, в систему поступает новая гипотеза в виде текста естественного языка, которая переводится в выражение логики предикатов. Для поиска аргументации строится дерево от заданного выражения на основе чистой логики предикатов. Этот метод позволяет находить аргументацию и контраргументы. Среди проблем алгоритмов, применяющих классическую логику можно отметить:

1. Громоздкость процесса вывода аргументации;
2. Процесс вывода и представления аргументации может быть не похож на рассуждения, с которыми работает человек;
3. Не учитывает, что в процессе аргументации может возникать неопределенность;
4. Классическая логика не позволяет получить новые знания, она приводит уже известные знания к нужной форме или проверяет возможность такого вывода.

Другим популярным подходом является применение нечетких логик. Например, экспертные системы с применением нечетких логик и алгоритма rete [2]. В процессе его работы к аргументам применяются правила вывода, при этом на выходе формируется направленный ациклический граф аргументации. Данный подход имеет тот же недостаток, что и алгоритмы, применяющие классическую логику, он не позволяет получать новые знания, применяет правила вывода для получения новой формулировки уже известного. Преимуществом нечетких логик является учет неопределенности в процессе аргументации.

Во многих работу на тему «argument mining» используются нейронные сети, что позволяет активно применять такие решения на практике. Например, программный продукт «Targer» [3] для извлечения аргументации из текстов естественного языка. В нейронных сетях нет сложности вывода классической логики, затраты на работу экспертов минимальны, работа проводится в автоматическом режиме, но понять причину того или иного вывода нейронной сети не представляется возможным, при этом трудно оценить, отражается ли семантика при выводе аргументации.

Таким образом, на данный момент не существует широко применяемой теории, которая бы могла одновременно и быть близкой к человеческим рассуждениям, и отражать семантику текстов на естественном языке, и решать проблему омонимии, и учитывать неопределенность, и получать новые знания в процессе вывода аргументации.

Теория, разрабатываемая в рамках данной работы, базируется на теории И. А. Мельчука «Смысл - Текст», в частности направлении, заданном в работах Д.Е.Пальчунова и Е.О.Ненашевой [4; 5]. В рамках данного подхода выражение на естественном языке можно рассматривать в виде специальной конструкции – ситуации, которая строятся на основе фрагментов атомарных диаграмм. Подход способен учитывать семантику предложения, его контекст, решать проблему омонимии, отображать связи между предложениями.

В процессе работы над темой были изучены существующие способы формализации знаний, развито направление представления знаний с помощью ситуаций и шаблонов, разработаны алгоритмы поиска знаний, сравнения знаний, получения знаний по аналогии с помощью описанных в работе схем, разработана и протестирована программная система на основе описанных подходов.

Разработанные методы и программная система призваны развить способы формального представления и сравнения знаний, предложить новые подходы к поиску аргументации.

Предложенные алгоритмы можно использовать для дальнейших разработок в теме формализации знаний, написанная программная система может применяться в качестве рекомендательной системы для поиска аргументации.

1 Описание предметной области

1.1 Понятие аргумента и аргументации

Аргумент – 1. Основание, доводы, приводимые в подтверждение или опровержение чего-либо. 2. Матем. Независимая переменная величина. [6]

В другом определении [19], близком к математической логике, аргумент – суждение, посредством которого обосновывается истинность другого суждения, состоящее из посылок и вывода.

Аргумент является ключевым понятием данной работы, его определение в терминах ситуаций в рамках разрабатываемой теории будет изложено позже.

Аргументация – 1. Способ, метод доказательства с помощью аргументов. 2. Совокупность аргументов. [6]

1.2 Знания и их представление

Знание – 1. Обобщённые представления о свойствах объективного мира, связях и отношениях между объектами, хранящиеся и воспроизводимые в сознании; осведомлённость в чём-либо. 2. Совокупность сведений, познаний в какой-либо области человеческой деятельности; обладание сведениями о чём-л. 3. Система сведений о закономерностях развития природы, общества и т. п . [6]

Онтология – это спецификация смысла, определения ключевых понятий, которыми описывается данная предметная область. [4]

Ранее упоминалось, что одним из недостатков классической логики является то, что с ее помощью нельзя получить новых знаний, только изложить известные факты в новой форме. Новые знания являются вероятностными. Они не выводятся явно из известных знаний без каких-либо допущений. Гипотезу можно считать новым знанием.

Выделим некоторые важные категории знаний, которые пригодятся в процессе работы над темой: гипотезы, пресуппозиции, гипотезы пресуппозиций.

1.3 Пресуппозиция

Пресуппозиция в лингвистической семантике — необходимый семантический компонент, обеспечивающий наличие смысла в утверждении.

Пресуппозиция может явно содержаться в предложении, а может подразумеваться. Пресуппозиция выражает общие знания. Хоть и это определение не является формальным, но оно соответствует тому, как человек воспринимает тексты естественного языка. Существует семантическая и синтаксическая пресуппозиции, в работе будем рассматривать семантическую.

Пресуппозиция может носить разный смысл, например: пресуппозиция синонимии, пресуппозиция обычностей, пресуппозиция взаимосвязей и т.д. В зависимости от этого смысла осуществляются различные переходы при получении гипотез и гипотез пресуппозиций.

1.4 Представление знаний с помощью фрагментов атомарных диаграмм

Рассматриваются модели вида: $\mathfrak{A} =< A;\ \sigma >=< A;\ P_1; \dots;\ P_n,\ c_1, \dots, c_m >$, где $\sigma =< P_1; \dots;\ P_n,\ c_1, \dots, c_m >$ - сигнатура, A – универсум модели, $P_1; \dots;\ P_n$ - предикатные символы, $c_1, \dots, c_m$- константные символы. Множество предложений сигнатуры – $S(\sigma)$. Модели описывают предметные области, позволяя дополнять сигнатуру при необходимости, и изучаются в развивающемся направлении теории моделей предметных областей.

Предложение φ атомарно, когда $\varphi = P(c_1, \dots, c_n)$, где $P, c_1, \dots, c_n \in \sigma$. Атомарная диаграмма модели $AD(\mathfrak{A}) = \{\varphi \in S(\sigma) |\ \mathfrak{A}_A \implies \varphi, \varphi$ – атомарное или его отрицание$\}$. Подмножество AD – фрагмент атомарной диаграммы. Конечное подмножество AD – конечный фрагмент атомарной диаграммы. Знания в данном случае являются частичными, неполными. Таким образом, имея предложение φ мы говорим о его истинности, имея отрицание φ говорим о его ложности, иначе значение φ на модели неизвестно.

Определения атомарной диаграммы используются в работах, описывающих методы извлечения и представления знаний из текстов естественного языка [4; 5; 7; 8].

Теория И.А. Мельчука позволяет представить глагол в виде многоместного предиката, так как глаголы имеют свойство вступать в связи с другими словами предложения. Это свойство называется валентностью. Глаголы, деепричастия и причастия представляются в виде n-местных предикатов с помощью программы LogicText [7], которая строит фрагменты атомарных диаграмма модели по текстам естественного языка. Далее, n-местные предикаты преобразуются к двухместным предикатам, это позволяет учитывать меняющиеся валентности глаголов. Наборы двухместных предикатов можно преобразовать к специальной структуре – ситуациям, которые будут описаны в дальнейших разделах.

2 Методическая часть

2.1 Выбор текстов для исследований

Для целей исследования возможностей применения разработанных методов необходимы тексты на естественном языке, в которых содержится много аргументации. Желательно, чтобы тексты были небольшими и принадлежали определенной тематике.

В качестве источника таких текстов были выбраны популярные новостные ресурсы, предоставляющие новости по определенным тематикам.

В целях автоматизации извлечения данных выбирались ресурсы, предоставляющие скачивание набора последних опубликованных текстов в определенной тематике с помощью протокола RSS.

2.2 Выделение аргументов в текстах

Выделение аргументации в текстах естественного языка – является обширной задачей, заслуживающей отдельных исследований. Для упрощения работы из текста выделяются только аргументы, содержащие маркеры аргументации.

Маркерами аргументации являются специальные слова или словосочетания, такие как «потому что», «поскольку», «так как». В русском языке зависимую часть сложноподчиненного предложения называют причинными придаточными. В качестве маркеров выделены наиболее популярные союзы, употребляющиеся с ними.

Таким образом, аргумент в самом общем виде имеет структуру «A -> B», где «->» - маркер аргументации, который означает следствие одной части утверждения из другой. Приведем несколько примеров аргументов, извлеченных из новостных статей:

А. [Также врач рассказала, что аллергический ответ возникает на масла в кожице мандарина]. **Поэтому**, [если человеку предложить уже очищенный мандарин, части аллергических реакций можно будет избежать].

Б. [Крупные рыбодобытчики могут отказаться от вылова сардины иваси, самой дешевой в России рыбы], **поскольку** [этот бизнес может оказаться нерентабельным в следующем году].

2.3 Ситуация

Ситуация – структура, предназначенная для формального представления предложения, которая состоит из:

1. Множества двухместных предикатов, при этом каждый из предикатов имеет вид P(S, c), где P - вопрос к глаголу, S – ситуация, в составе которой находится предикат, c – слово, извлеченное из текста, являющаяся ответом на вопрос к глаголу.

2. Множества подситуаций, являющихся ситуациями. Например, несколько грамматических основ могут описываться разными подситуациями.

3. Ситуация может быть под отрицанием. В самом простом случае отрицание означает отрицание над предикатом.

В базовом случае ситуация содержит только двухместные предикаты, среди которых обязательно есть предикат «Что сделать (S, глагол_действие)», и соответствует предложению с одной грамматической основой.

Формально, ситуация S – ациклический граф, в листьях которого содержатся двухместные предикаты $\varphi_i \in AD(\mathfrak{A}_S)$, где $i \in [0, n]$, n - общее число двухместных предикатов для S, $AD(\mathfrak{A}_S)$ - фрагмент атомарной диаграммы для S. Вершины этого графа могут содержать отрицания.

Плюсом данного подхода является то, что имея некоторые знания, описываемые фрагментом атомарной диаграммы $AD(\mathfrak{A})$, их можно дополнить новыми знаниями $AD(\mathfrak{A}_S)$, даже если у двух указанных диаграмм не совпадают сигнатуры. Добавление информации возможно при условии, что новые знания не противоречат старым, то есть что не существует $\varphi \in AD(\mathfrak{A}_S)$ и $[\text{не}]\varphi \in AD(\mathfrak{A})$ или $[\text{не}]\varphi \in AD(\mathfrak{A}_S)$ и $\varphi \in AD(\mathfrak{A})$, где φ – многоместный предикат, соответствующий ситуации или подситуации.

2.4 Преобразование предложения к ситуации

Для преобразования предложений естественного языка к набору ситуаций необходимо выполнить следующие шаги:

1. Трансформация предложения:
   a. Исключение глаголов, означающих вероятность действия;
   b. Замена глаголов-отрицаний на частицу “не”;
2. Переход от предложения на естественном языке к множеству многоместных предикатов;
3. Замена словосочетания с прилагательным на дополнительные простые предложения с глаголом “являться”;
4. Указание констант-ситуаций;
5. Преобразование каждого многоместного предиката к набору двухместных;
6. Непосредственно преобразование к ситуации.

Опишем подробнее указанные шаги и то, почему их необходимо выполнить.

2.5 Трансформация предложения

Для более правильного понимания предложения возможно в ручном или автоматизированном режиме преобразовать его.

Перечислим основные правила трансформации. В первую очередь, необходимо исключить модальности. Модальность – глаголы, означающие вероятность какого-либо другого глагола, такие как «может».

Следующим шагом будет преобразование к отрицанию негативных глаголов, сочетающихся с другими глаголами. Например, словосочетание «перестать вылавливать» можно преобразовать в «не вылавливать». Отрицания выносятся перед многоместными предикатами, а в дальнейшем и перед соответствующей ситуацией.

Для автоматического преобразования необходимо составить список всех возможных глаголов, означающих отрицание, а также список глаголов-модальностей. После этапа синтаксического разбора предложения (определения частей речи, отношений зависимости между словами) выполнять трансформацию предложения.

2.6 Получение многоместных предикатов

Как уже упоминалось, для текста естественного языка можно получить наборы n-местных предикатов. Программная система Logic Text позволяет получить набор фрагментов атомарных диаграмм, которые и являются многоместными предикатами. При этом места одного предиката имеют различные типы. У каждого глагола свой набор типов предикатов. В различных предложениях один и тот же глагол может иметь различные наборы типов предикатов. Опишем переход от многоместных предикатов к ситуациям.

Приведем пример многоместного предиката с указанием типов и мест предиката для предложения «*Вчера контейнеровоз встал поперек Суэцкого канала»*.

Фрагменты атомарных диаграмм для предложения: *Встал(что: контейнеровоз, где: поперек Суэцкого канала, когда: вчера)*

2.7 Преобразование многоместных предикатов к двухместным

Для обработки знаний с использованием технологии Semantic Web с помощью языка описания онтологий OWL, основанного на RDF, необходимы двухместные предикаты. Отчасти поэтому в целях удобства формального представления ситуаций, используются двухместные предикаты. В данной работе не будет использоваться OWL, но его применение могло бы быть направлением для дальнейшей работы по теме.

Для перехода к ситуации следующим шагом является преобразование многоместных предикатов к наборам двухместных. Это осуществляется путем преобразования n-местного предиката в набор из n+1 двухместных предикатов.

В работе Д.Е.Пальчунова и Е.О.Ненашевой [5] подробно описано преобразование многоместных предикатов к наборам двухместных, кратко изложим суть метода. Необходимо привести предложение к следующему виду:

1. В предложении только одна грамматическая основа, присутствует только один глагол;
2. В предложении присутствуют только существительные, наречия, числительные, предлоги;
3. В предложении только одно прилагательное. Прилагательное Y допускается только в предложениях вида «X является Y».

Для удовлетворения указанных требований необходимо:

1. Сложносочиненные предложения и предложения с однородными членами разбиты на несколько простых предложений;
2. Причастия и деепричастия заменены на соответствующие глаголы;
3. Местоимения заменены на существительные или прилагательные, на которые они ссылаются;
4. Словосочетания с прилагательным заменены на дополнительные простые предложения.

Результат программы LogicText удовлетворяет пунктам 1, 2 и 3. Последний пункт реализован в написанной для данной работы программной системе.

Рассмотрим пример для предложения «*Гигантский контейнеровоз развернулся и встал поперек Суэцкого канала*».

Соответствующий набор приведенных предикатов:

Развернуться(развернуться_0, контейнеровоз)

Являться(гигантский, контейнеровоз)

Встать(встать_0, контейнеровоз, поперек Суэцкого канала)

Следующим шагом необходимо обозначить константы-ситуации. Константа-ситуация является численно-буквенным идентификатором ситуации. С каждым глаголом ассоциируется своя ситуация.

Далее, необходимо преобразовать многоместные предикаты к наборам двухместных. Так как место предиката имеет тип, к каждому из них можно задать вопрос, ответом на который будет соответствующее значения. Получаем наборы двухместных предикатов вида $P(A_i, c)$, где P – вопрос к слову; A_i - обозначение константы-ситуации, соответствующей данному утверждению; c – константа.

Например, перейдем от констант-действий к константам-ситуациям для предыдущего примера:

Развернуться(S_1, развернуться_0, контейнеровоз)

Являться(S_2, гигантский, контейнеровоз)

Встать(S_3, встать_0, контейнеровоз, поперек Суэцкого канала)

Здесь S_1, S_2, S_3 – константы-ситуации, введенные для формального представления приведенного предложения.

Далее, преобразуем многоместный предикат к набору двухместных для Встать(S_3, встать_0, контейнеровоз, поперек Суэцкого канала):

Что делать(S_3, встать)

Что(S_3, контейнеровоз)

Где(S_3, , поперек Суэцкого канала)

2.8 Переход к ситуации

После получения для каждого глагола, причастия и деепричастия наборов двухместных предикатов, необходимо объединить их в ситуации. Каждый набор двухместных предикатов, описывающих отдельный глагол, причастие, деепричастие объединяется в отдельную ситуацию.

Например, для примера в предыдущем пункте «*Гигантский контейнеровоз развернулся и встал поперек Суэцкого канала»* составим ситуацию (рисунок 1):

Что делать(S_1*, развернуться)*

Что(S_1*, контейнеровоз)*

Где(S_1*, поперек Суэцкого канала)*

Что делать(S_2*, являться)*

Каким(S_2*, гигантский)*

Что(S_2*, контейнеровоз)*

Что делать(S_3*, встать)*

Что(S_3*, контейнеровоз)*

Где(S_3*, поперек Суэцкого канала)*

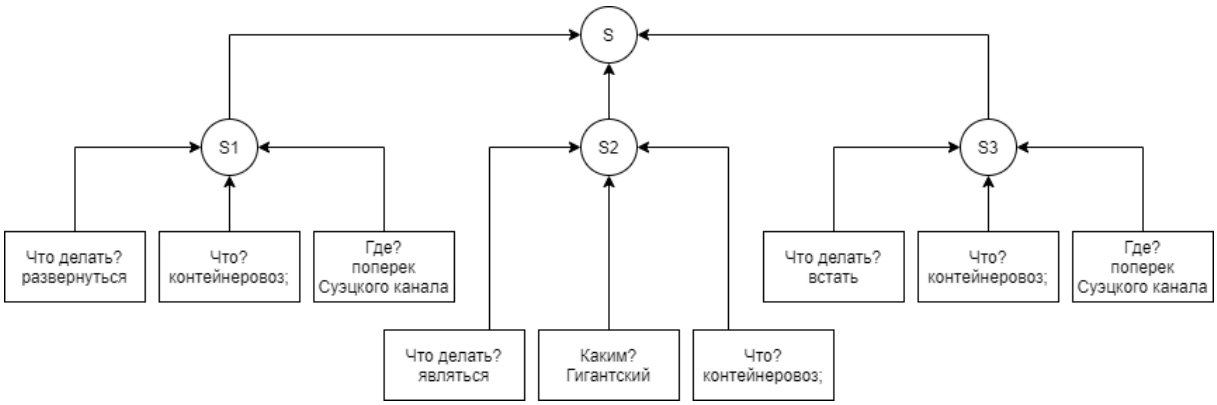


Рисунок 1 — Пример ситуации, составленной по предложению

2.9 Представление аргумента с помощью ситуации

Сформулируем формальное определение аргумента с использованием ситуаций. Аргументом назовем такую структуру $[\text{не}]S_{k_1}, \dots, [\text{не}]S_{k_n}, [\text{не}]S_{p_1}, \dots, [\text{не}]S_{p_v} \rightarrow [\text{не}]S_1, \dots, [\text{не}]S_m$, что существуют ситуации, $S_{k_1}, \dots, S_{k_n}$, $S_{p_1}, \dots, S_{p_v}$, $S_1, \dots, S_m$ удовлетворяющие ей, при этом не существует $S_{k_1}', \dots, S_{k_n}', S_{p_1}', \dots, S_{p_v}'$ таких что $kn' < kn$ и $pv' < pv$.

В определении $S_{k_1}, \dots, S_{k_n}$ содержатся явно в утверждении, а $S_{p_1}, \dots, S_{p_v}$ находятся в составе пресуппозиции. Эти знания могут явно содержаться в тексте, или же подразумеваться автором.

В данной работе рассматриваются утверждения без кванторов.

Аргументы извлекаются из рассматриваемого текста. Гипотезой-аргументом назовем структуру, аналогичную аргументу, но она не извлекается из текста, а получается с помощью разработанных в данной работе схем вывода аргументов-гипотез. Также существует гипотеза-ситуация, которая получается из схем вывода ситуаций-гипотез.

В работах на тему моделирования аргументации на основе теории И. А. Мельчука «Смысл - Текст», например в работе [8], было введено понятие шаблона ситуации и шаблона рассуждений. С учетом определенного понятия аргумента с использованием ситуаций можно дань определения шаблонов.

Шаблон ситуации – набор ситуаций, соответствующий утверждению естественного языка.

Шаблон рассуждения – структура, состоящая из двух частей, содержащих ситуации, связанных причинно-следственной связью (маркером аргументации). Будем считать, что шаблон рассуждений – то же самое, что аргумент с использованием ситуаций.

В соответствие с введенными определениями одной из главных целей формализации знаний с помощью ситуаций и шаблонов является разработка единой автоматической процедуры поиска аргументов по шаблонам и генерации на их основе новых знаний.

2.10 Рассуждение по аналогии

Рассуждение по аналогии – правдоподобное рассуждение, соответствующее тому, как человек аргументирует свою позицию, он пытается найти сходство между новым утверждением и уже известными знаниями.

Выделяют две разновидности аналогий [10]: по свойствам, когда два сравниваемых объекта имеют сходство по выделенным свойствам; по отношениям, когда объекты схожи в рамках определенного отношения.

В работе будет рассматриваться аналогия отношений. Ключевым отношением между аргументами является отношение сходства. Гипотеза схожа с аргументом, если ее вывод осуществляется с помощью схем получения гипотез-аргументов. Схемы будут рассмотрены далее.

Введем отношение инцидентности [21; 22]. Пусть имеем фрагмент атомарной диаграммы $\mathrm{AD}(\mathfrak{A})$ и $\varphi \in \mathrm{S}(\sigma)$. $\mathrm{AD}(\mathfrak{A}) \,||\, \varphi$ если $\mathrm{AD}(\mathfrak{A}) \cup \{\varphi\} \nvdash$. Это определение является формальным описанием возможности существования ситуации, описанной $\mathrm{AD}(\mathfrak{A})$, на которой выполняется φ.

Гипотеза является знанием, о точной истинности или ложности которого нам неизвестно. Пусть имеем $\mathrm{AD}(\mathfrak{A})$ и гипотезу, которая описывается набором $\varphi_1 \in \mathrm{S}(\sigma), \ldots, \varphi_N \in \mathrm{S}(\sigma)$. Минимальным требованием к гипотезе является, чтобы $\varphi_1 \,||\, \mathrm{AD}(\mathfrak{A})$, …, $\varphi_N \,||\, \mathrm{AD}(\mathfrak{A})$.

2.11 Отношения между ситуациями

Отношения являются важнейшим понятием в рассуждениях, так как сами по себе ситуации не представляют большой ценности, гораздо важнее то, как они связаны, чтобы в ходе аргументации было возможно применить аргумент, аналогичный известному.

Будем рассматривать отношения, отражающие семантику сравниваемых выражений. Чем более похожи ситуации, тем более они схожи семантически. Точное равенство является частным случаем семантического сходства. Различные типы отношений между понятиями в онтологиях дают различные типы отношений между ситуациями.

Формально, имеем ситуацию S_1, описываемую в рамках атомарной диаграммы $AD(\mathfrak{U}_{S_1})$, а также S_2, описываемую $AD(\mathfrak{U}_{S_2})$. Для сравнения сходства ситуаций необходимо предложить функцию $F(S_1, S_2)$ в рамках $AD(\mathfrak{U}) = AD(\mathfrak{U}_{S_1}) \cup AD(\mathfrak{U}_{S_2})$.

Так как существует несколько вариантов для сравнения двух ситуаций, выбирается тот вариант, который бы дал максимальное сходство.

Например, пусть ситуация S_1 состоит из двухместных предикатов $P = \{p_1, \ldots, p_n\}$, ситуация S_2 состоит из предикатов $B = \{b_1, \ldots, b_n\}$. Чтобы определить сходство $d(S_1, S_2)$ необходимо подобрать такое отображение $F: P \to B$, чтобы $\sum_{i=1}^{n} d(p_i, F(p_i))$ была максимальна.

Приведем пример, на основе которого будут рассматриваться отношения между ситуациями.

«*Крупные рыбодобытчики могут отказаться от вылова сардины иваси, самой дешевой в России рыбы*»

Сопоставим данному предложению набор многоместных предикатов {являться(S_1, кто: рыбодобытчик, каким: крупный, НЕвылавливатьS1, кто: рыбодобытчик, что:сардина иваси, являться(S1_4, каким:самый дешевый, объект:рыба)}.

Также, сопоставим ему ситуацию на рисунке 2.

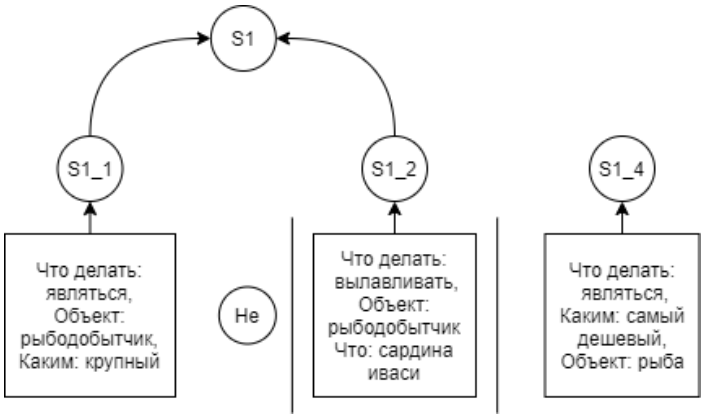


Рисунок 2 — Ситуации S_1 и S_{1_4}, составленные по предложению из примера

2.11.1 Сходство понятий в онтологии

В общем случае для измерения расстояния между понятиями в онтологии, вводится метрика [9], зависящая от числа перегибов и длины пути между понятиями в древовидном представлении множества понятий. Понятия считаются схожими, если соединены коротким путем с малым числом перегибов. Перегибы бывают двух видов: перегиб сверху, когда сначала понятие обобщается, а затем уточнятся, а также перегиб снизу, когда понятие сначала уточняет, а затем обобщается. Условно говоря, можно строить пути вверх-вниз по дереву онтологий.

Описанное семантическое расстояние между понятиями можно использовать для сравнения ситуаций, а именно для предикатов в ситуациях. Для сравнения двух двухместных предикатов их типы предикатов должны быть равны.

2.11.2 Неразличимые с точностью до некоторой степени подобия ситуации

Для начала рассмотрим отношения для ситуаций с полным пересечением, то есть когда ни в одной из сравниваемых ситуаций не исключается какая-либо ситуация или предикат. Нужно отметить, что для такого сравнения необходимо, чтобы ситуации имели одинаковую структуру, одинаковое число предикатов и подситуаций.

Ситуации неразличимы, если их подситуации и предикаты неразличимы (очень схожи) или в точности равны. Частным примером такого отношения являются ситуации, состоящие из синонимичных предикатов.

Приведем пример ситуации S_3 на рисунке 3, схожей с S_1 с точностью до всех предикатов, которые являются равными или неразличимыми. S_3 соответствует предложению: «Крупные добытчики рыбы могут перестать вылавливать сардину иваси». Переход осуществлен за счет заменены понятия «рыбодобытчик» на «добытчик рыбы» и «отказаться» на «перестать».

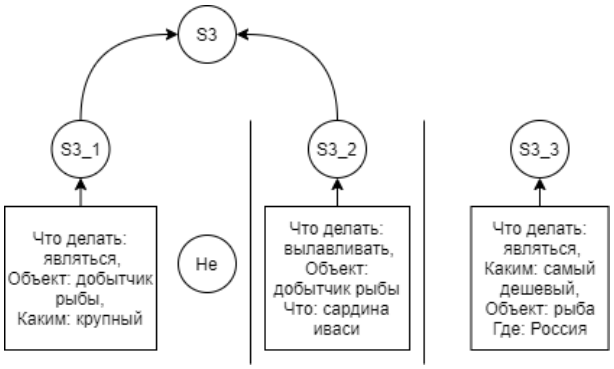


Рисунок 3 — Ситуации S_3 и S_{3_4}

Ранее упоминалось, что при сравнении может возникнуть несколько вариантов, среди которых выбирается тот, что дает наибольшее сходство. Рассмотрим на примере сравнения ситуаций S_1и S_3. Существует 2 варианта попарно сравнениями все элементы множества $\{S_{1\_1}, S_{1\_2}\}$ и все элементы $\{S_{3\_1}, S_{3\_2}\}$. Первый сравнивает S_{1_1} с S_{3_1} и S_{1_2} с S_{3_2}. Второй сравнивает S_{1_1} с S_{3_2} и S_{1_2} с S_{3_1}.

2.11.3 Общее-частное

Продолжим рассматривать отношения с полным пересечением и рассмотрим важный частный случай пункта 2.11.2. Одна ситуация является обобщением другой, если

значения предикатов одной ситуации строго общие по иерархии в онтологии над значениями предикатов другой ситуации при равных типах предикатов.

Например, ситуация S_4 на рисунке 4 для предложения «Крупные рыбодобытчики могут отказаться от вылова Sardinops sagax sagax» является частной к S_1. Sardinops sagax sagax является подвидом сельди иваси.

Ситуация S_5 на рисунке 5 для предложения: «Крупные рыбодобытчики могут отказаться от вылова сардины» является общей над S_1. Переход осуществлен за счет замены понятия «сардина иваси» на понятие «сардина».

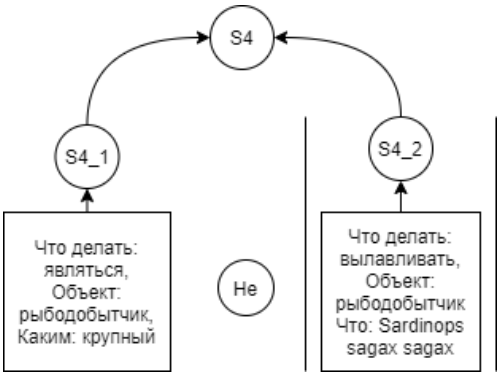


Рисунок 4 — Ситуация S_4

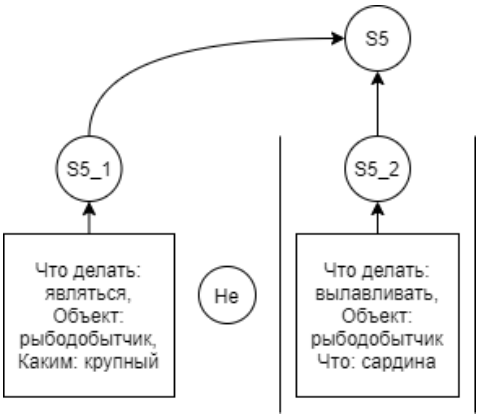


Рисунок 5 — Ситуации S_5

2.11.4 Пересечение

При пересечении ситуации содержат общие схожие части. Для такого типа сравнения допускается выкидывать из рассмотрения часть подситуаций или предикатов в сравниваемых ситуациях, после чего можно проверить сходство получившихся ситуаций. Этот подход является наиболее общим подходом к подбору схожей ситуации.

Частным случаем является расширение ситуации. В данном отношении одна ситуация является частью другой или содержит очень похожую часть. Например, ситуация S_6 на рисунке 6, составленная для предложения: «Крупные рыбодобытчики,

находящиеся в Мурманске, могут отказаться от вылова сардины иваси», является расширением S_1 за счет добавления информации о местонахождении рыбодобытчиков (добавлена подситуация S_{6_2}).

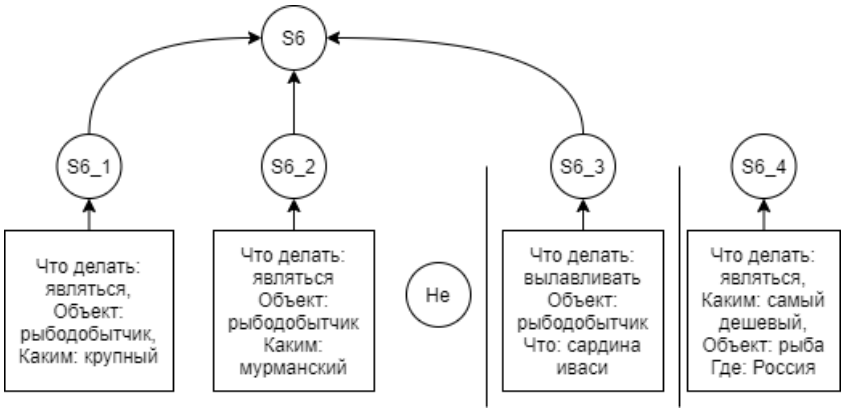


Рисунок 6 — Ситуации S_6 и S_{6_4}

Другой частный случай – усечение ситуации. Приведем пример ситуации S_7 на рисунке 7, составленной для предложения: «Рыбодобытчики могут отказаться от вылова сардины иваси». S_7 является усечением S_1. В данном случае исключена информация о том, что рыбодобытчики являются крупными.

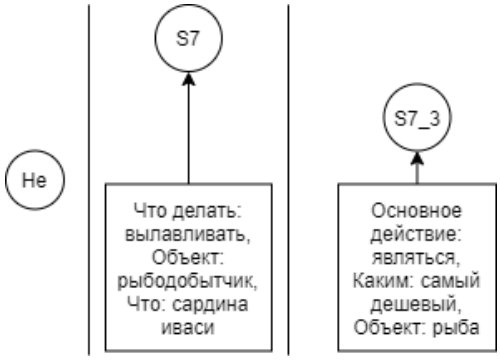


Рисунок 7 — Ситуации S_7 и S_{7_4}

Приведем пример более общего отношения пересечения. Ситуация S_8 на рисунке 8 составлена по предложению: «Крупные рыбодобытчики, находящиеся в Мурманске, могут перестать вылавливать Sardinops sagax sagax». Для получения S_8 добавили к S_1 подситуцию S_{8_5}, убрали несущественную подситуацию S_{1_5}, в S_{1_3} заменили понятие “отказаться” на схожее понятие “перестать”, в ситуации S_{1_4} понятие “сардина иваси” на частное понятие “Sardinops sagax sagax”.

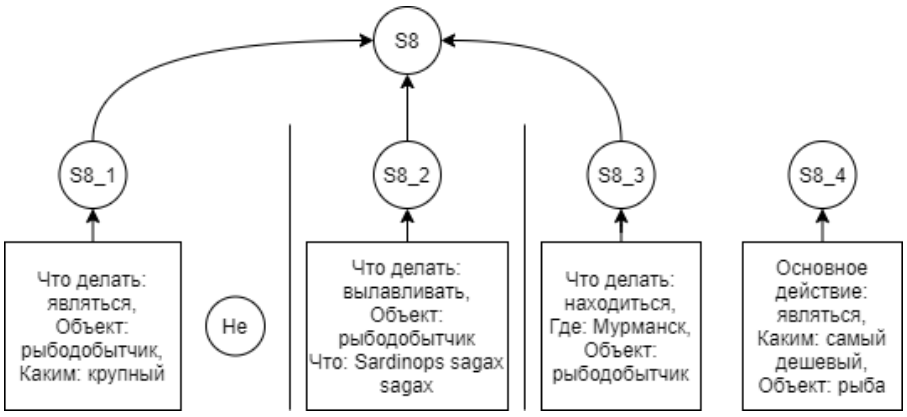


Рисунок 8 — Ситуация S_8

2.11.5 Сходство по набору свойств

Если в онтологии для понятий присутствуют атрибуты, то можно учитывать близость их атрибутов или близость атрибутов родительских понятий. Таким образом, при сравнении двух ситуаций можно сравнивать их предикаты по сходства атрибутов понятий.

2.11.6 Раскрытие понятия через ситуацию

Для такого типа отношений понятие в предикате заменяется его описанием в виде ситуации.

Например, в ситуации S_1 заменить понятие «сельдь иваси» на его определение в виде ситуации S_{9_2}. Таким образом, получается S_9, соответствующая предложению: «*Крупные рыбодобытчики могут отказаться от вылова [дальневосточной сардины, которая с советских времен известна как «сельдь иваси»](* S_{9_2}*), самой дешевой в России рыбы*»

2.11.7 Сравнение понятий с учетом тематики

Для сравнения понятий в онтологии можно учитывать гранулированность кластеров [9]. Выделяется основная тематика с главным кластером, остальные являются фоновыми и расстояния вычисляются относительно основного кластера. В частности, можно сравнивать понятия из двух разных онтологий с помощью величины, основанной на числе мостов между объединением двух онтологий. Например, есть две онтологии, узко специализирующиеся на определенных тематиках, их понятия можно сравнить.

2.12 Схемы преобразования шаблонов

Схемы преобразования шаблонов позволяют получать на основе уже имеющихся знаний некоторые гипотезы-аргументы или гипотезы-ситуации. Данный подход дает

возможность осуществлять рассуждение по аналогии. В рамках работы над темой был разработан ряд схем получения гипотез.

Также, можно составлять композиции из схем, применяя одну схему за другой, получая гипотезы от гипотез.

Имея шаблон ситуации A и атомарную диаграмму $\mathrm{AD}(\mathfrak{U})$, схема преобразования $\mathrm{R}(\mathrm{A})$ получает гипотезу, описываемую набором $\varphi_1 \in \mathrm{S}'(\sigma), \ldots, \varphi_\mathrm{N} \in \mathrm{S}'(\sigma)$, а также может дополнить $\mathrm{AD}(\mathfrak{U})$ до новой атомарной диаграммы $\mathrm{AD}'(\mathfrak{U})$. После преобразования необходимо проверить, чтобы $\varphi_1 \,||\, \mathrm{AD}'(\mathfrak{A})$, …, $\varphi_\mathrm{N} \,||\, \mathrm{AD}'(\mathfrak{A})$.

2.12.1 Замена ситуации в условии аргумента на схожую гипотезу

В данной схеме ситуации в условии аргумента заменяются на схожие или в точности равные. Некоторые ситуации можно исключить.

Например, пусть имеем аргумент: [Крупные рыбодобытчики перестанут вылавливать сардину иваси](S_1), [самую дешевую в России рыбы](S_{1_4}), **поскольку** [этот бизнес может оказаться нерентабельным в следующем году.](S_2)

a. Заменим S_1 ситуацией S_3 путем замены понятия «рыбодобытчик» на «добытчик рыбы». Получим гипотезу: [Крупные добытчики рыбы перестанут вылавливать сардину иваси](S_3), [самую дешевую в России рыбу](S_{3_4}), **поскольку** [этот бизнес может оказаться нерентабельным в следующем году.](S_2)

b. Другим частным случаем является подбор общих или частных ситуаций. Например, заменим ситуацию S_1 на S_4. Переход осуществлен за счет замены понятия «сардина» на название ее подвида «Sardinops sagax sagax».

Получим гипотезу: [Крупные рыбодобытчики перестанут вылавливать Sardinops sagax sagax](S_4), [самую дешевую в России рыбы](S_{4_4}), **поскольку** [этот бизнес может оказаться нерентабельным в следующем году.](S_4)

c. В более общем случае можем заменять ситуации из условия аргумента на схожие пересекающиеся ситуации. Приведем несколько примеров.

Заменим ситуацию S_1 на S_6 за счет добавления информации о том, что рыбодобытчики из Мурманска.

Получим гипотезу: [Крупные рыбодобытчики, находящиеся в Мурманске, перестанут вылавливать сардину иваси](S_6) , [самую дешевую в России рыбу](S_{6_4}), **поскольку** [этот бизнес может оказаться нерентабельным в следующем году.](S_2)

d. Другим частным примером является усечение ситуации, а также удаление каких-либо ситуаций полностью из условия или вывода аргумента. Хорошими

кандидатами для удаления являются ситуации из пресуппозиции, не являющиеся важными в процессе аргументации. Для примера на рисунке 4.а можно удалить ситуацию S_{1_4} из пресуппозиции, так как она не является существенной.

Таким образом, получим гипотезу: [Крупные рыбодобытчики перестанут вылавливать сардину иваси](S_1), **поскольку** [этот бизнес может оказаться нерентабельным в следующем году](S_2).

е. Приведем более общий пример с заменой S_1 на S_8. Получим аргумент: [Крупные рыбодобытчики, находящиеся в Мурманске, перестанут вылавливать Sardinops sagax sagax](S_8), **поскольку** [этот бизнес может оказаться нерентабельным в следующем году.](S_2)

2.12.2 Замена ситуации в выводе аргумента на схожую гипотезу

Аналогичным предыдущему пункту образом можно заменять ситуации в выводе аргумента на схожие.

2.12.3 Дополнение ситуации

Дополнять ситуации можно тремя способами:

1. Дополнять ситуации новыми знаниями. Для этого необходимо добавить к исходной ситуации новую подситуацию, предикат или добавить к набору ситуаций в условии или выводе аргумента новую ситуацию с учетом существующей сигнатуры атомарной диаграммы.
2. Расширение сигнатуры атомарной диаграммы, добавить новое свойство уже существующих элементов.
3. Дополнение новыми знаниями методами логического вывода. В таком случае новой информации нет, перерабатываем известные знания. Схемы логического вывода можно получить путем проведения аналогии с существующими формулами классической логики.

2.12.4 Силлогизм

Силлогизм — вид дедуктивного умозаключения, состоящего из двух посылок и одного заключения. Является методом логического вывода. В рамках силлогизма есть три термина: большой, малый, средний.

Требования к терминам и посылками:

1. Средний термин должен присутствовать хотя бы в одной из посылок
2. Термин, не присутствующий в посылке, должен быть в заключении

3. Должна быть хотя бы одна общая посылка

4. Если одна из посылок частная, то заключение тоже частное

5. Должна быть хотя бы одна утвердительная посылка, то есть все посылки не могут быть отрицательными

6. Число отрицательных посылок равно числу отрицательных заключений.

Формализуем категорический силлогизм для подхода с использованием ситуаций. На рисунке 9 изображена общая схема силлогизма, S – подситуация, соответствующая меньшему термину, P — подситуация, соответствующая большему термину, M — подситуация, соответствующая среднему термину. Имея ситуации S_1 и S_2 можем получить ситуацию S_3 , при этом S_2 – знания из пресуппозиции, имеющие тип пресуппозиции сходства (то есть имеющие структуру по типу «A – это B»).

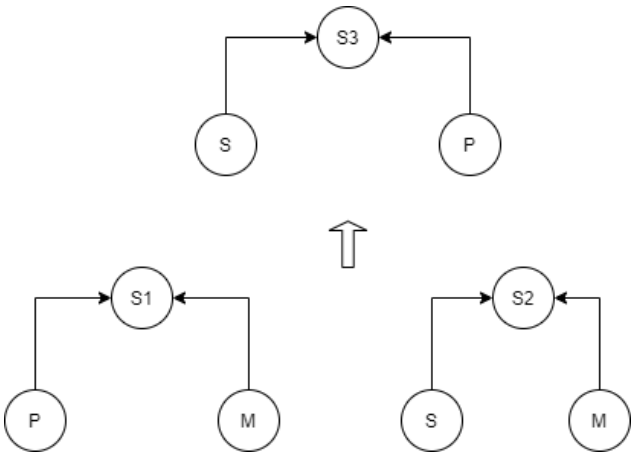
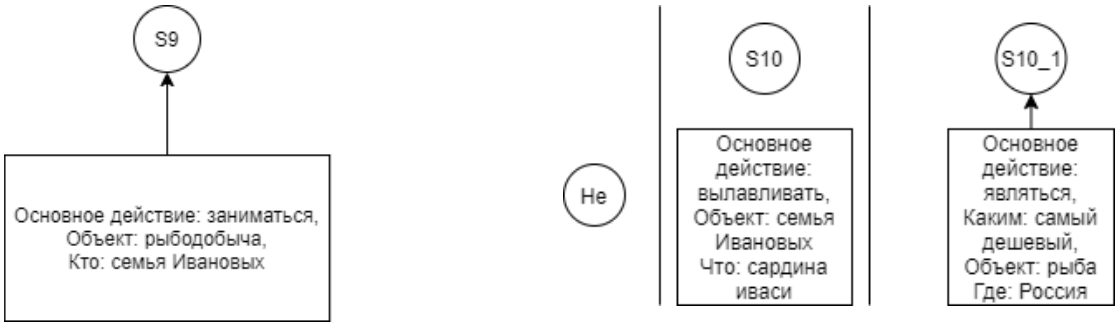


Рисунок 9 — Общая схема силлогизма

Приведем пример. Пусть имеем ситуацию S_1 , изображенную на рисунке 2. В соответствии со схемой она содержит большую и среднюю подситуации. Также, пусть имеем S_9 на рисунке 10.а, соответствующую выражению «Семья Ивановых занимается рыбодобычей». В ней содержатся малая и средняя подситуации. С помощью схемы силлогизма можно получить S_{10} на рисунке 10.б, соответствующую выражению «Семья Ивановых откажется от вылова сардины иваси, самой дешевой в России рыбы».



а. б.

Рисунок 10 — Ситуации S_9 (а) и S_{10} (б)

Таким образом, заменив в аргументе S_1 на S_9, получаем гипотезу: [Семья Ивановых перестанет вылавливать сардину иваси](S_9), **поскольку** [этот бизнес может оказаться нерентабельным в следующем году.](S_2).

2.12.5 Аргументация от эксперта

Пусть существует предложение, содержащее информацию о том, что некий эксперт высказался о чем-либо. В рамках данной схемы можем откинуть часть предложения, соответствующую информации о том, что высказывание принадлежит эксперту, тем самым получим непосредственную суть предложения. Общая схема приведена на рисунке 11. Ситуация S содержит множества подситуаций E и P, при этом E соответствует части предложения о том, что эксперт высказывается, а P соответствует непосредственной сути высказывания. E содержит предикат, схожий по смыслу с «основное_действие(E, сказать)».

Например, пусть имеем аргумент: [[Эксперт утверждает](E), что [крупные рыбодобытчики перестанут вылавливать сардину иваси](P)](S), **поскольку** [этот бизнес может оказаться нерентабельным в следующем году.](S_2). Ситуации для его условия изображены на рисунке 12. Применяя схему для S, получим гипотезу: [Крупные рыбодобытчики перестанут вылавливать сардину иваси](P), **поскольку** [этот бизнес может оказаться нерентабельным в следующем году.](S_2)

Данную схему можно расширить. Переход будет состоять из двух шагов. На первом шаге знаем, что A утверждает о чем-либо из области знаний K. С помощью знаний из пресуппозиции необходимо вывести, что A является экспертом в области M, при этом K является подобластью M. A является экспертом, если принадлежит какой-либо группе экспертов, о которой мы знаем. В примере есть группа экспертов «руководство рыбодобывающей компании», а есть его представитель «заместитель директора в рыбодобывающей компании». Вторым шагом необходимо откинуть информацию об авторе высказывания, оставив только суть высказывания.

Например, имеем несколько утверждений:

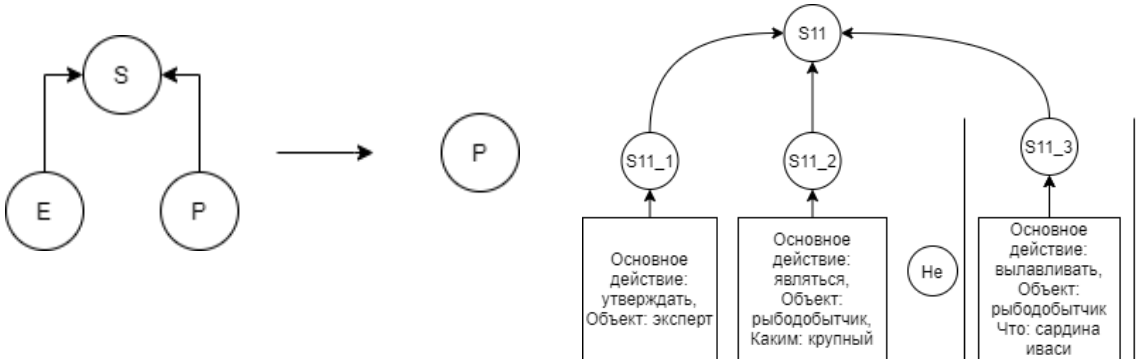
1. [Человек, работающий в руководстве рыбодобывающей компании, знает, откажутся ли рыболовы от вылова рыбы иваси](S_{11}).

2. [Иван работает заместителем директора в рыбодобывающей компании](S_{12}).

3. [Иван утверждает, что крупные рыбодобытчики откажутся от вылова сардины иваси](S_{14}).

Получим гипотезу: [Иван знает, будут ли рыболовы вылавливать сардину иваси](S_{13}), **поскольку** [Иван работает заместителем директора в рыболовной компании](S_{12}) и [человек, работающий в руководстве рыбодобывающей компании, знает, откажутся ли рыболовы от вылова рыбы иваси](S_{11}).

Далее, можем получить: [Рыболовы откажутся от вылова рыбы иваси](S_{15}), **поскольку** [Иван знает, будут ли рыболовы вылавливать сардину иваси](S_{13}) и [Иван утверждает, что крупные рыбодобытчики откажутся от вылова сардины иваси](S_{14}).



а. б.

Рисунок 11 — Общая схема аргументации от эксперта

2.12.6 Объединение условий и выводов

Пусть имеем два аргумента. Если имеем утверждение, содержащее условие и первого, и второго аргумента, то можем получить гипотезу, содержащую в условии условия первого и второго аргумента, а в выводе выводы первого и второго аргумента.

Приведем пример. Имеем аргументы:

1. [Крупные рыбодобытчики перестанут вылавливать сардину иваси](S_1), [самую дешевую в России рыбу](S_{1_4}), **поскольку** [этот бизнес может оказаться нерентабельным в следующем году.](S_2)
2. [Вылов лосося может стать прибыльным](S_{16}), **так как** [на него наблюдается высокий спрос](S_{17}).
3. [Бизнес по вылову сардины иваси может оказаться нерентабельным, и наблюдается высокий спрос на лосось.](S_{18})

Объединив вывод, можем получить следующую гипотезу: [Крупные рыбодобытчики могут отказаться от вылова сардины иваси и начать вылавливать

лосось](S_{19}), **поскольку** [бизнес по вылову сардины иваси может оказаться нерентабельным, и наблюдается высокий спрос на лосось.](S_{18}).

2.12.7 Схема с нахождением аналогичного

Пусть имеется ситуация, которая говорит о том, что подситуация A имеет одинаковые черты с подситуацией B. Например, она может содержать прилагательное, схожее слову «одинаковый». Также имеем ситуацию C, схожую с A. Можем заменить в общей ситуации подситуацию A на C. Схема приведена на рисунке 12. По сути это является частным случаем замены ситуации на схожую, но с учетом структуры выражения можно получить дополнительную информацию.

Приведем пример. Имеем выражения:

1. [Иван и я работаем в одной рыбодобывающей компании](S_{20}).
2. [Люди, работающие в одной рыбодобывающей компании, имеют одинаковые знания о способах добычи той или иной рыбы](S_{21}).

Получим гипотезу: [Иван и я имеем одинаковые знания о способах добычи сардины иваси](S_{22}), **поскольку** [Иван и я работаем в одной рыбодобывающей компании](S_{20}) и [люди, работающие в одной рыбодобывающей компании, имеют одинаковые знания о способах добычи той или иной рыбы](S_{21}).

Далее, имея выражение «Я знаю, что рыболовы перестанут вылавливать рыбу иваси», можно сделать вывод, что «Иван знает, что рыболовы перестанут вылавливать рыбу иваси».

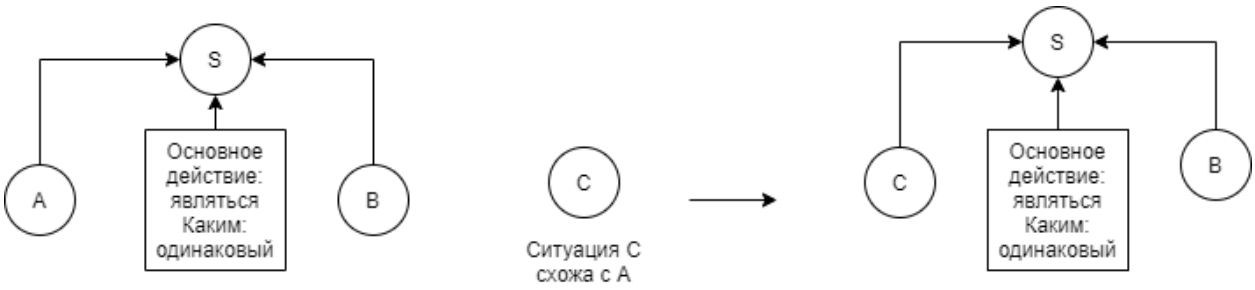


Рисунок 12 — Общая схема ситуации с нахождением аналогичного

2.13 Итог методической части

Таким образом, на основе теории Мельчука «Смысл-текст» и работ в направлении формализации текстов естественного языка описан алгоритм представления аргументов с помощью ситуаций.

Важной частью работы являются разработанные отношения между ситуациями и методы сравнения ситуаций. Отношения полагаются на семантику ситуаций за счет

использования онтологий. При этом работа фокусируется на схожих с некоторой точностью ситуациях.

Для поиска аргументации применяется рассуждение по аналогии, в рамках которого для нахождения доказательства или вывода из какого-либо утверждения получить гипотезу и найти имеющийся схожий аргумент.

Нахождение гипотез, аналогичных аргументам, и получение схожих ситуаций осуществляется с помощью разработанных схем, которые опираются на отношения сходства между ситуациями.

3 Программная реализация

Для разработанных подходов к представлению аргументов естественного языка, нахождения аргументации, сравнения ситуаций, схем получения гипотез была разработана программная система.

Программная система позволяет представлять тексты естественного языка в виде набора ситуаций, извлекать аргументы, сохранять их в виде структуры с ситуациями, получать аргументацию для заданных утверждений с помощью применения схем получения гипотез. Use case диаграмма для пользователя системы представлена на рисунке 13. Предусмотрена одна роль для пользователя.

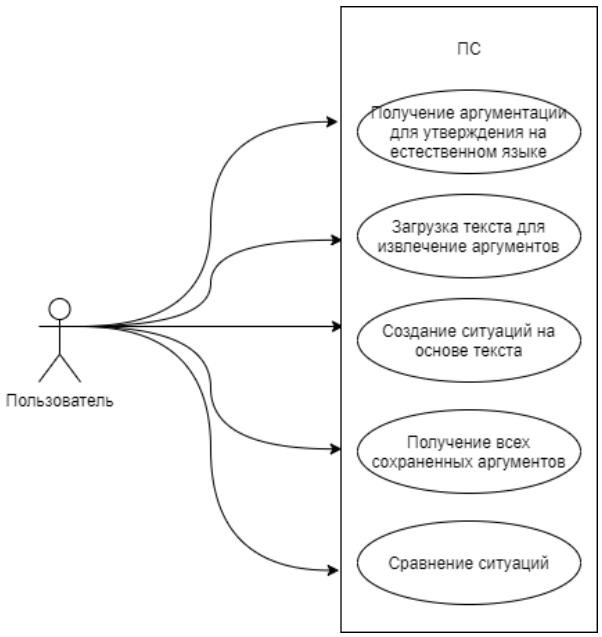


Рисунок 13 — Use case диаграмма для программной системы

3.1 Описание программной системы

Программная система состоит из: серверной части, реализующей основную логику; графического приложения для взаимодействия с пользователем и серверной частью. Схема работы программной системы приведена на рисунке 14. Серверная часть состоит из: основного сервера, написанного на java с помощью фреймворка Spring; сервера для взаимодействия с онтологиями, написанного на языке Python.

Основной сервер взаимодействует с программной системой LogicText [7] для получения фрагментов атомарных диаграмм из текстов на естественном языке.

Серверная часть для взаимодействия с онтологиями использует библиотеку wiki-ru-wordnet для использования API словаря WikiWordnet [11].

Для пользователя написано графическое приложение на java на основе библиотеки JavaFX [12].

Подробности о процессе установки и использования программы приведены в приложении А.

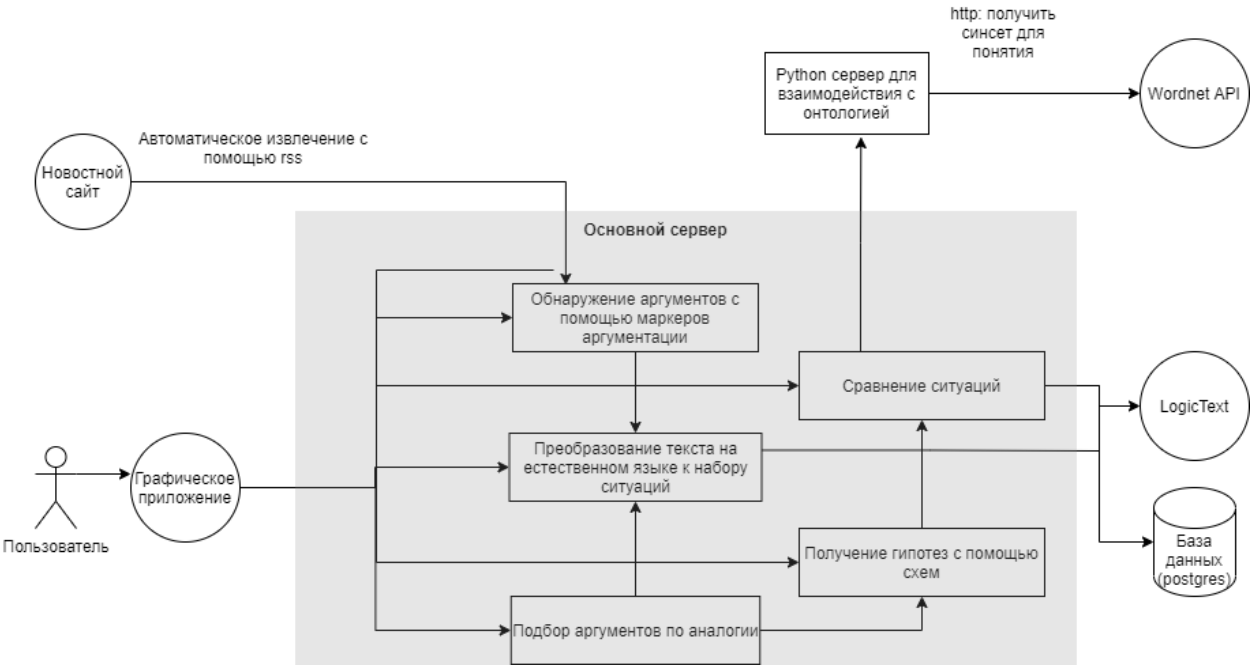


Рисунок 14 — Схема программной системы

3.2 Исходные данные для извлечения аргументов

Новостные тексты для извлечения аргументов могут быть получены системой двумя способами: автоматически и в ручном режиме. В ручном режиме пользователь может загрузить текст с помощью графического приложения, поместив текст непосредственно в форму для ввода, или выбрав файл с текстом в файловой системе компьютера.

В автоматическом режиме скрипт выкачивает актуальные новостные статьи с сайта mk.ru по протоколу RSS и далее извлекает из выкачанных текстов аргументы.

3.3 Основной сервер

В основном сервере реализован следующий функционал:

1. Загрузка текста для поиска и сохранения аргументов. Загрузка возможна через запросы на API сервера;
2. Извлечение аргументов по маркерам в тексте;
3. Взаимодействие с LogicText;
4. Взаимодействие с API сервера онтологий;

5. Реализация представления ситуаций, аргументов с помощью ситуаций. UML диаграмма классов приведена на рисунке 15;
6. Преобразование текстов к набору ситуаций;
7. Сравнение ситуаций;
8. Поиск аргументации для утверждений естественного языка с использованием схем получения гипотез;
9. Взаимодействие с СУБД. Использовалась СУБД Postgres [13] и фреймворк Hibernate [14] для взаимодействия;
10. Получение всех сохраненных аргументов;
11. Предоставление конечных точек для взаимодействия с основным сервером. Подробная информация о списке контрольных точек представлена в описании программы.

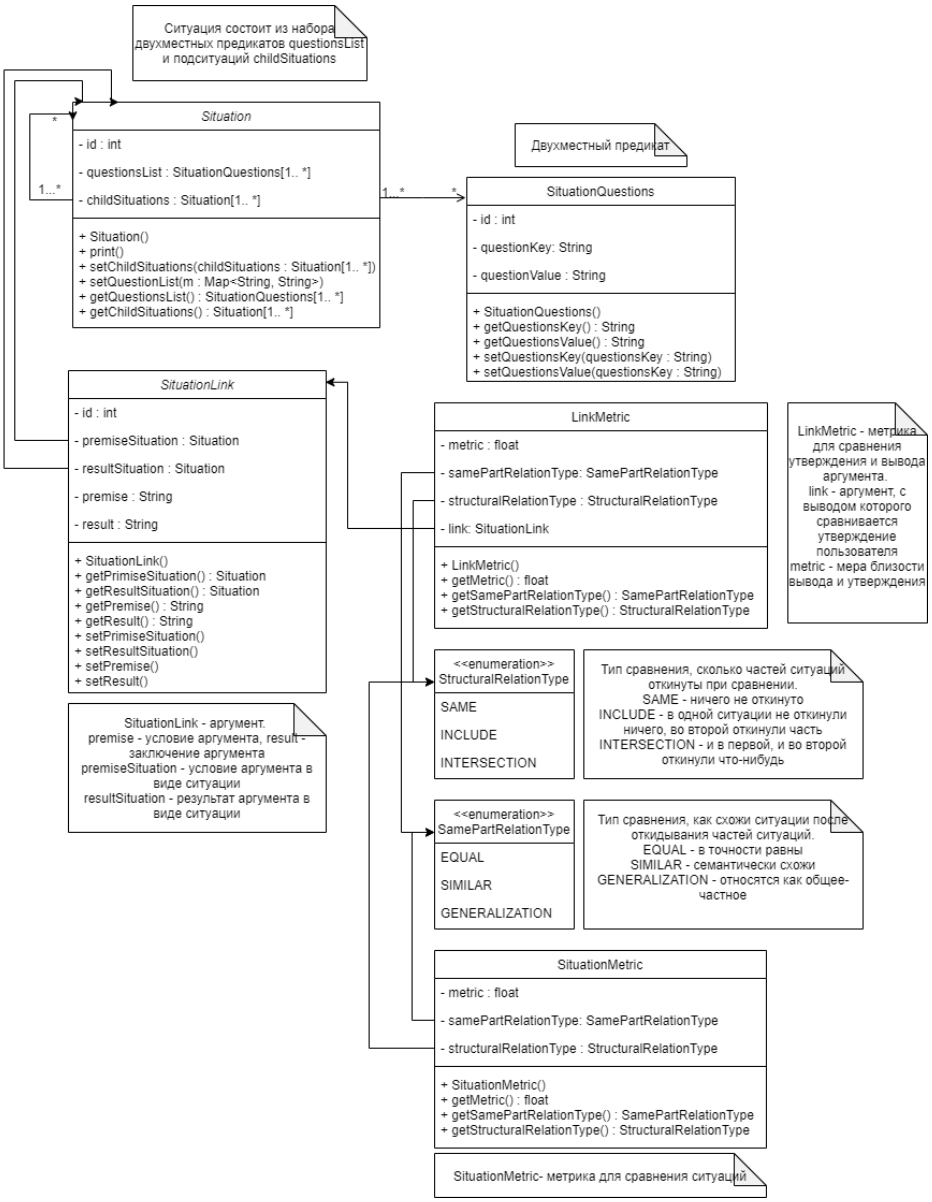


Рисунок 15 — UML диаграмма основных классов

3.4 Извлечение аргументов

Аргументы извлекаются из новостных текстов с помощью маркеров аргументации, описанных в главе 2.2. Извлечение происходит путем проверки, содержатся ли маркер аргументации в каждом предложении текста.

Полный список маркеров аргументации, отобранных для программной системы: «благодаря тому что», «в результате того что», «в связи с тем что», «в силу того что», «ввиду того что», «вследствие того что», «затем что», «значит», «поэтому», «потому», «следовательно», «ибо», «из-за того что», «оттого что», «по причине», «поскольку», «потому что», «так как», «тем более что».

Для каждого маркера обозначено направление аргументации, означающее, как посылка и заключение аргумента располагаются относительно маркера. Если в предложении обнаружен маркер аргументации, предложение разбивается на часть до и после маркера. В зависимости от направления аргументации на маркере, заполняются поля посылки и заключения аргумента в специальной структуре, которая сохраняется в СУБД. Также перед сохранением посылка и заключение из утверждений на естественном языке преобразуются к ситуациям.

3.5 Интеграция с системой извлечения аргументов

Фокусом данной работы в первую очередь является формализация и обработка утверждений естественного языка с целью генерации новых знаний и поиска аргументации. Распознавание и извлечение аргументов является комплексной задачей, мной реализован лишь самый базовый метод с использованием маркеров аргументации. Для улучшения качества аргументов необходимо интегрироваться с существующей системой, способной решать задачу их извлечения из текстов.

В работе [18] разработана система, способная получать аргументы из текста естественного языка, для этого необходимо отправить запрос на сервер. Ответом является список структур, изображенных на рисунке 16. В ней содержится посылка, заключение аргумента, а также тип аргумента. Аргумент может быть «за» посылку или «против», то есть являться контраргументом посылки. Интеграция с описанной системой не реализована на данный момент.

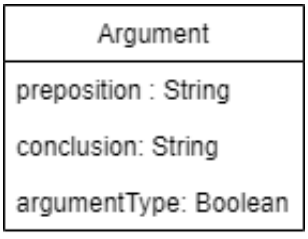


Рисунок 16 — Структура извлеченного аргумента

3.6 Интеграция с системой LogicText

Программная система LogicText предназначена для получения математических моделей над текстами естественного языка. В рамках данной работы был необходим реализованный в системе функционал по построению фрагментов атомарных диаграмм для текстов. На основе фрагментов атомарных диаграмм (наборов предикатов) строятся ситуации.

Для взаимодействия была собрана из исходного кода и подключена библиотека LogicText. Также в рамках основного сервера написан сервис, позволяющий вызывать метод по обработке текста из библиотеки, получать из ответа метода библиотеки набор предикатов и разрешенные ссылки из местоимений для последующей замены.

3.7 Преобразование предложения естественного языка к ситуациям

Алгоритм преобразования утверждений естественного языка описан в главе 2.4. Трансформация предложения является необязательными пунктом, он призван улучшить работу алгоритма, в программной системе не реализован.

Преобразование текста в набор многоместных предикатов реализовано программной системой LogicText, с которой взаимодействует сервер. Далее, на основе вывода LogicText происходит замена местоимений на слова, на которые они ссылаются.

Из вывода LogicText можно выделить предикаты, содержащие прилагательные. Прилагательные необходимо выделить в отдельный предикат «Являться (Объект: слово1, Какой: слово2)».

Для обозначения констант-ситуаций необходимо для каждого глагола указать отдельную контсанту-ситуацию.

Далее, необходимо преобразовать многоместные предикаты к наборам двухместных. В отдельный двухместный предикат выделяется основной глагол и каждое место предиката.

Указанные шаги были реализованы в сервисе в рамках основного сервера.

3.8 Онтология

Для семантического сравнения понятий необходим онтологический словарь. Общий принцип сравнения понятий в иерархической онтологии описан в рамках главы 2.11.1. На данный момент бесплатных онтологий с таким детальным способом сравнения мной не найдено. В процессе работы было проведено сравнение существующих наиболее проработанных словарей онтологий, результаты приведены в таблице 1.

Таблица 1 — Сравнительная характеристика имеющихся словарей онтологий

Название	Поддерживаемые типы отношений между понятиями	Число понятий, тыс.	Статус	Способ программного взаимодействия
Викисловарь Wiki-word-net	синонимы, антонимы, гиперонимы, гипонимы, родственные слова	450	Открытый, свободный	Имеет модуль для взаимодействия на Python
Russian Distributional Thesaurus	Дистрибутивный тезаурус русского языка	932	Открытый, свободный	Нет
Yarn	Синонимы	15	Открытый, свободный	Есть исходный код, можно развернуть локальный сервер с помощью docker
BabelNet	синонимы, антонимы, гиперонимы, гипонимы, меронимы, определения	985	Открытый, несвободный	Есть открытое API, но бесплатно можно сделать не более 1000 запросов за сутки
РуТез	синонимы, антонимы, гиперонимы, гипонимы, меронимы	158	Открытый, несвободный	Нет
ABBYY Lingvo	синонимы, антонимы, родственные слова	Не указано	Закрытый	Есть, необходимо оплачивать программу

Среди представленных словарей отбираются открытые словари, имеющие способы программно взаимодействовать, поддержку для русского языка, отношения синонимии и родственных отношений общее-частное. На основе составленных требований выбран словарь Wiki-Word-Net.

Wiki-word-net позволяет получить для слова список синсетов. Синсет – это множество слов, объединенных схожим одним смыслом. Также есть методы для получения гиперонимов и гипонимов заданного синсета. Под гиперонимом понимается близкое по иерархии обобщающее слово. Под гипонимом понимается близкое по иерархии частное слово.

Для взаимодействия со словарем онтологий был написан отдельный HTTP сервер на языке python. Сервер использует официальный модуль wiki-ru-wordnet. Доступные на сервере методы описаны в описании программы. Пример запроса приведен на рисунке 17.

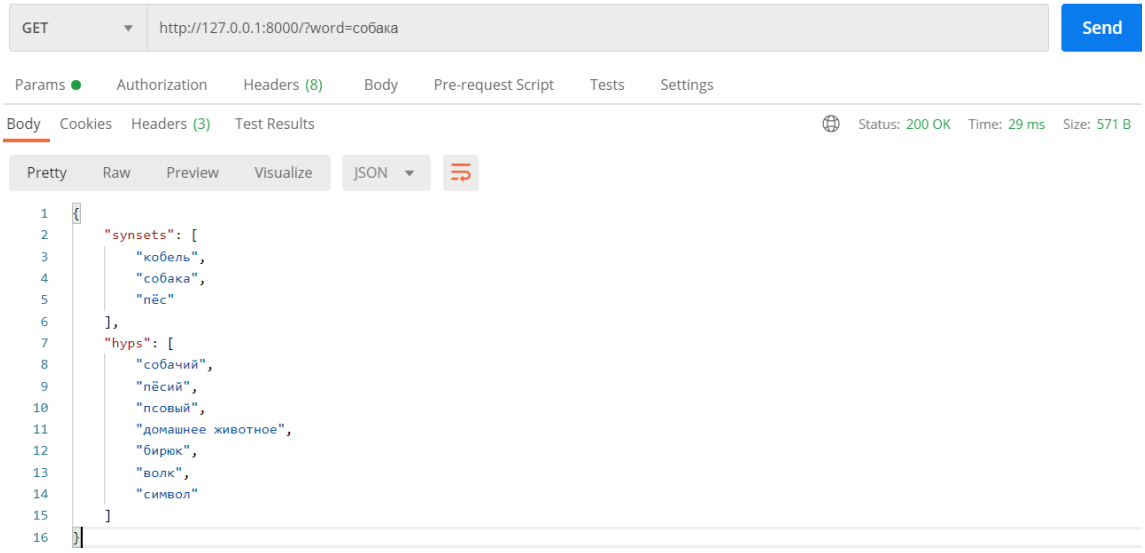


Рисунок 17 — Пример запроса к серверу онтологий

3.9 Сравнение ситуаций

В целях упрощения разработки приложения, реализованы не все описанные ранее описанные способы сравнения ситуаций и схемы получения гипотез, а только ключевые. Подробное описание алгоритмов для сравнения ситуаций находится в описании программы в приложении Б.

В программной системе ситуация представляется структурой Situation и состоит из:

1. Подситуаций;
2. Двухместных предикатов.

При этом преобразование к ситуациям устроено таким образом, что у ситуации может быть непустым только либо набор подситуаций, либо набор двухместных предикатов.

В соответствии со структурой ситуации разработано три алгоритма для разных вариантов сравнения ситуаций:

1. Обе ситуации содержат только подситуации;

2. Обе ситуации содержат только двухместные предикаты;
3. Одна ситуация содержит только подситуации, вторая содержит только двухместные предикаты.

Результатом сравнения является специальная структура, содержащая следующие поля для сравнения:

1. Численная мера сходства ситуаций;
2. Тип отношений, указывающий на то, будут ли при сравнении откинуты ситуации. Среди возможных значений:
    a. SAME – не исключена из рассмотрения ни одна из подситуаций и предикатов в обеих ситуациях;
    b. INCLUDE - в одной ситуации откинута часть подситуаций или предикатов, в другой ничего не откинуто;
    c. INTERSECTION – в обеих ситуациях было что-то откинуто.
3. Тип отношений, указывающий на то, в каком типе сходства находятся ситуации после откидывания каких-либо ситуаций. В частном случае ничего не было откинуто. Среди возможных значений:
    a. EQUAL – в точности равны;
    b. SIMILAR – семантически схожи, синонимичны;
    c. GENERAL – одна обобщает другую.

Как уже упоминалось в главе 2.11., при сравнении перебираются все возможные варианты сравнить подситуации и предикаты двух ситуаций. Результатом сравнения является метрика, дающая наибольшее численное значение сходства ситуаций. В процессе сравнения считается мера сходства и типы в соответствии с алгоритмом, изложенным в описании программы.

Для сравнения двух предикатов существует несколько вариантов:

1. Типы предикатов не равны;
2. Типы предикатов равны и значения аргументов предикатов в точности равны;
3. Типы предикатов равны и значения аргументов предикатов синонимичны;
4. Типы предикатов равны и одно значение аргумента предиката обобщает другое.

3.10 Схемы аргументации

Для аргументации используются описанные ранее схемы получения гипотез. Для того чтобы получить аргументацию, используется схема с заменой ситуаций в составе

результата аргумента на схожую ситуацию. Направлением для дальнейшей работы по теме является реализация и других описанных схем получения гипотез.

Для упрощения разработки программы в условии и заключении аргумента может содержаться только одна ситуация. Для обеспечения этого критерия ситуации для утверждения на естественном языке объединяются в одну для условия или вывода аргумента. В соответствии с этим схема с заменой ситуации в выводе вырождается в схему, где заменяется единственная в выводе ситуация на схожую.

Для настройки порога близости заменяемых ситуаций необходимо настроить численное значение, задающее нижнюю границу метрики близости (поле metric структуры LinkMetric) между ситуациями. При этом тип отношений между ситуациями не настраивается, при получении аргументации система предлагает все варианты, проходящие минимальный порог близости между ситуацией в выводе аргумента и ситуацией, построенной по утверждению пользователя.

3.11 Клиентское приложение

Клиентское приложение обеспечивает возможность пользователю получать доступ к основному функционалу системы через графическое приложение.

При запуске приложения открывается окно с возможностью получить аргументацию для утверждения пользователя и открыть другие окна. Интерфейс адаптируется к размеру окна приложения. Утверждение пользователя для поиска аргументации должно представлять сбой одно предложение, начинаться с большой буквы и заканчиваться точкой. Система выводит аргументы, для которых вывод схож с утверждением пользователя, при этом указывается численная мера близости и тип отношения. Условия и вывод аргумента даны в виде утверждений естественного языка.

На рисунке 18 приведен пример поиска аргументации. Поле «premise» соответствует условию аргумента, «result» заключению аргумента, «Same part similarity type» типу сходства после отбрасывания некоторых подситуаций и предикатов, «Structural similarity type» типу отбрасывания подситуаций и предикатов.

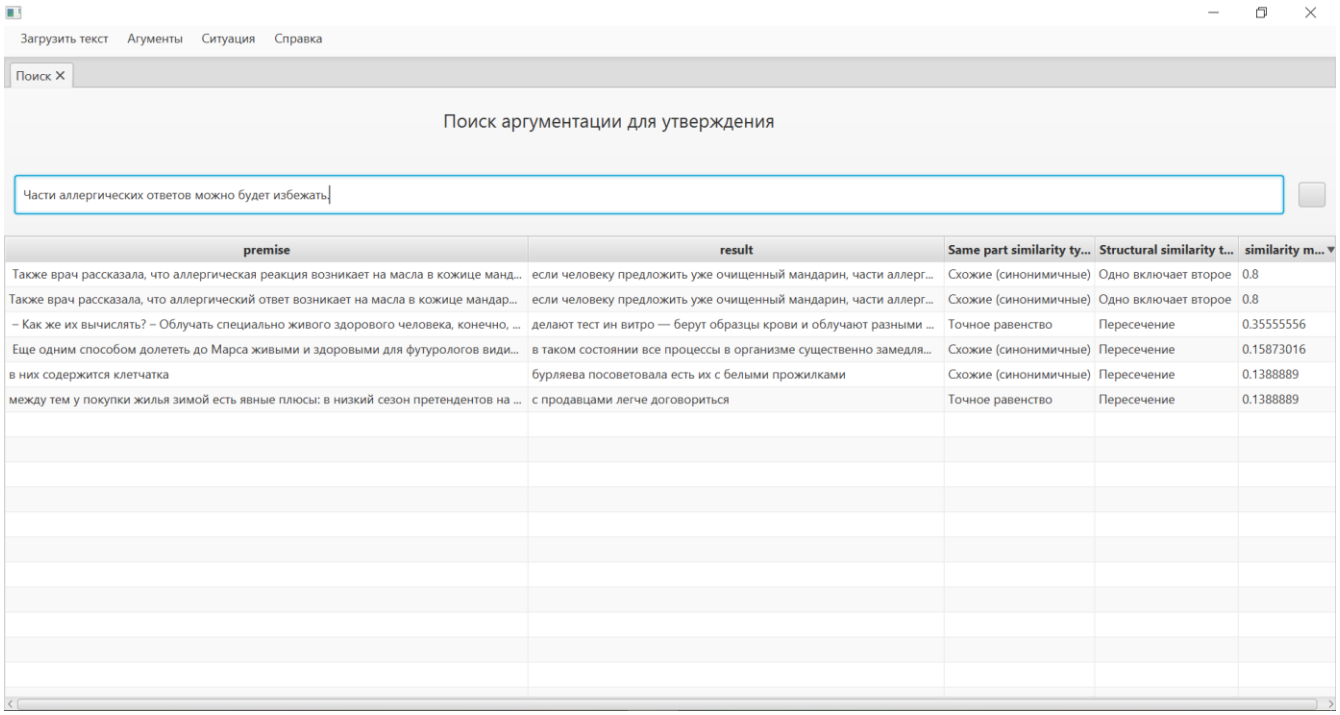


Рисунок 18 — Пример поиска аргументации для утверждения пользователя

Также, у пользователя есть возможность загрузить текст для извлечения аргументации несколькими способами: путем ввода текста для извлечения (рисунок 19); путем выбора .txt файла с текстом (рисунок 20); путем запуска скачивания последних опубликованных статей с новостного сайта mk.ru по теме «наука» (рисунок 21).

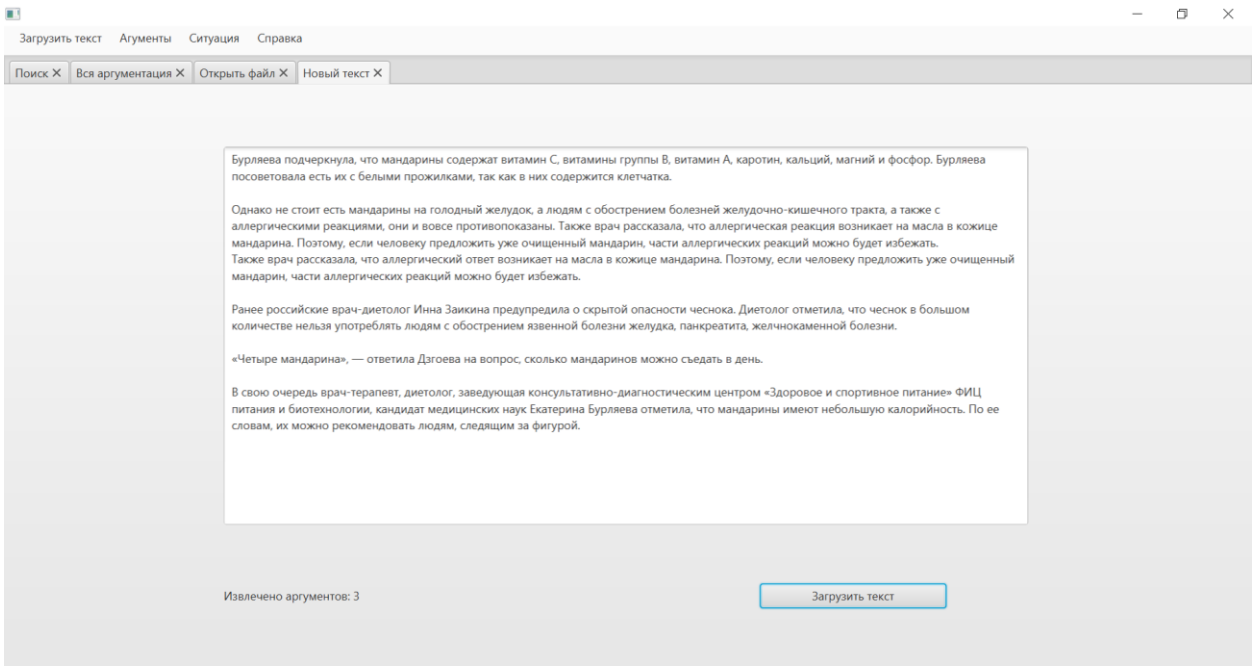


Рисунок 19 — Загрузка текста и результат загрузки текста

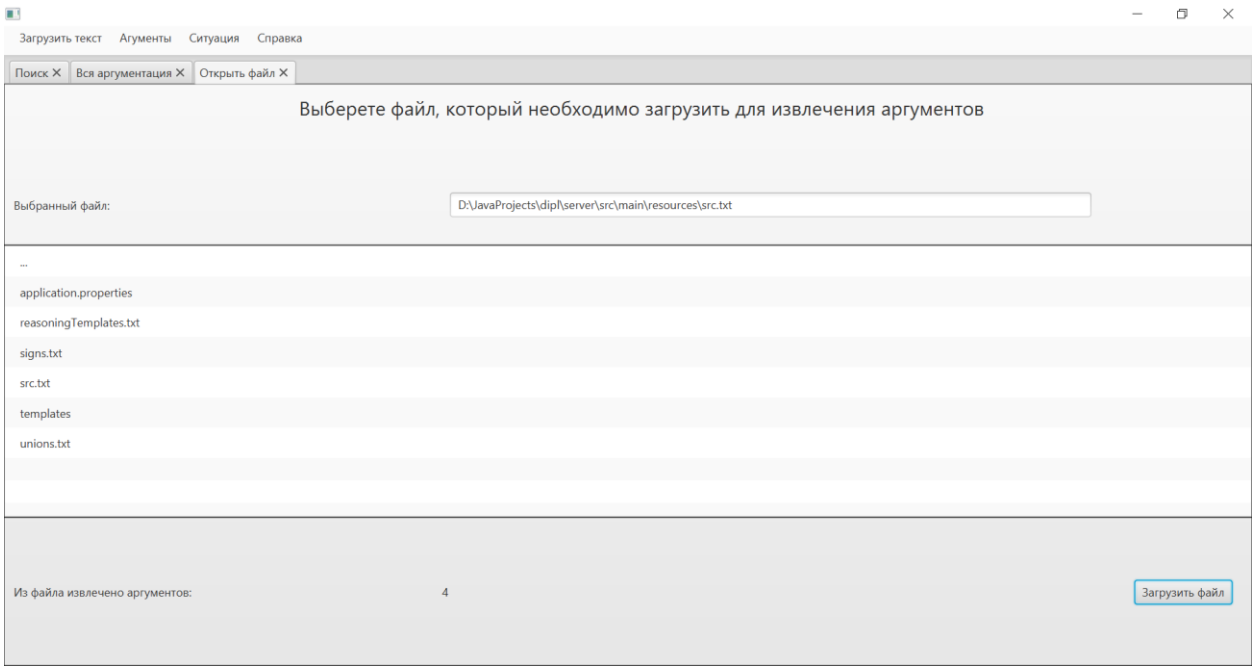


Рисунок 20 — Выбор файла и результат извлечения из файла

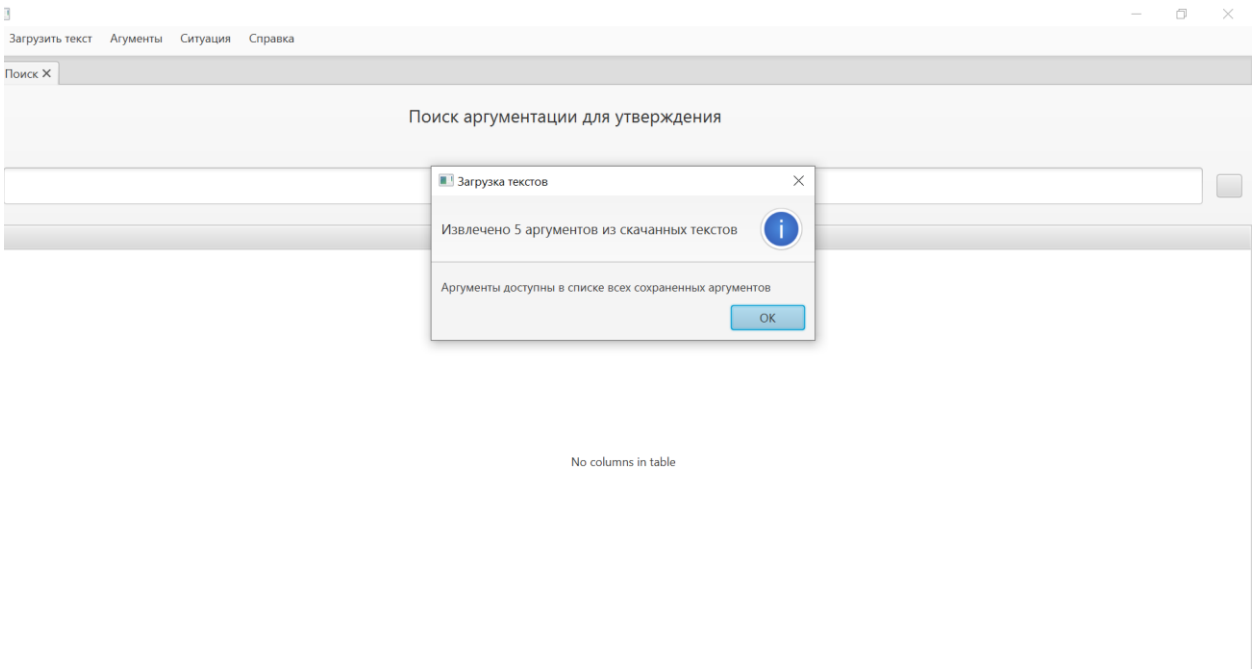


Рисунок 21 — Результат запуска скачивания новостных текстов

Пользователь имеет возможность просмотреть все сохраненные на данный момент аргументы. На рисунке 22 изображен пример, поле «premise» соответствует условию аргумента, поле «result» соответствует заключению аргумента.

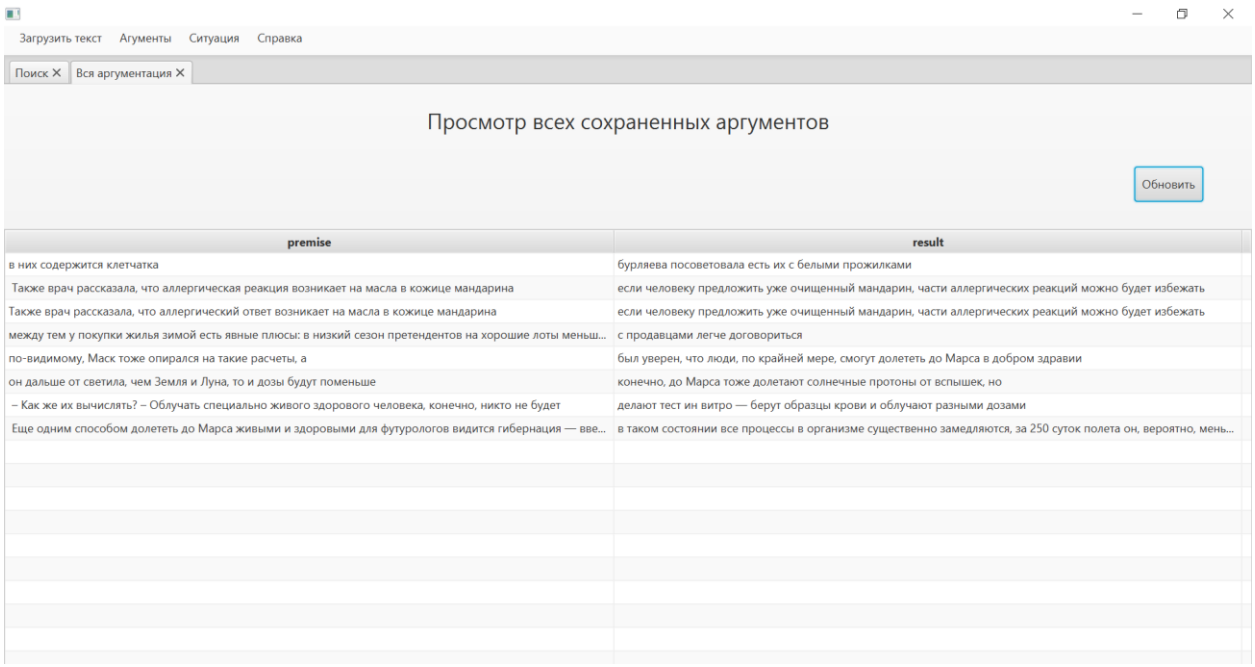


Рисунок 22 — Получение всех сохраненных аргументов

Пользователю доступен ряд инструментов для взаимодействия с ситуацией. Система предоставляет возможность преобразовать текст в набор ситуаций. Пример изображен на рисунке 23. В корне древовидного представления содержится пронумерованный список полученных по тексту ситуаций. Далее, для каждой ситуации рекурсивно указаны пронумерованные подситуации. В листьях находятся двухместные предикаты. До двоеточия указывается тип предиката, после двоеточия указывается значение предиката.

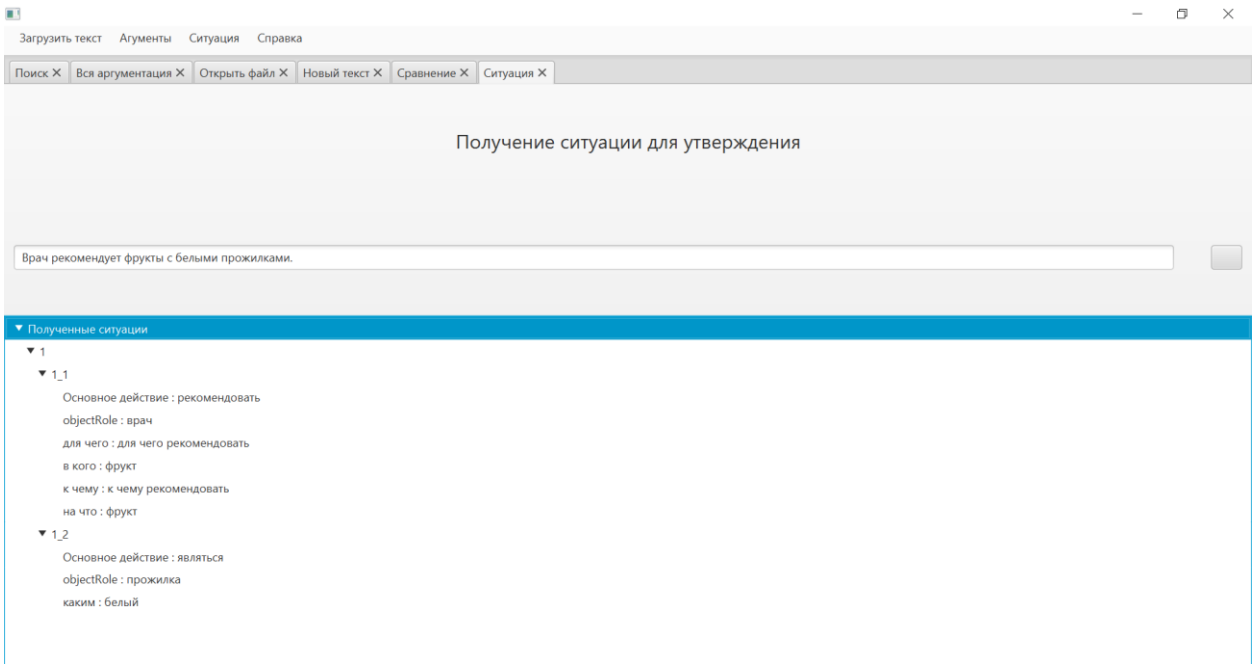


Рисунок 23 — Получение ситуации по тексту естественного языка

Система предоставляет возможность сравнения ситуаций, составленных по утверждениям естественного языка. На рисунке 24 изображен пример сравнения ситуаций. В качестве результата указываются вышеописанная метрика близости, тип откидывания частей ситуаций, тип сходства ситуаций после откидывания.

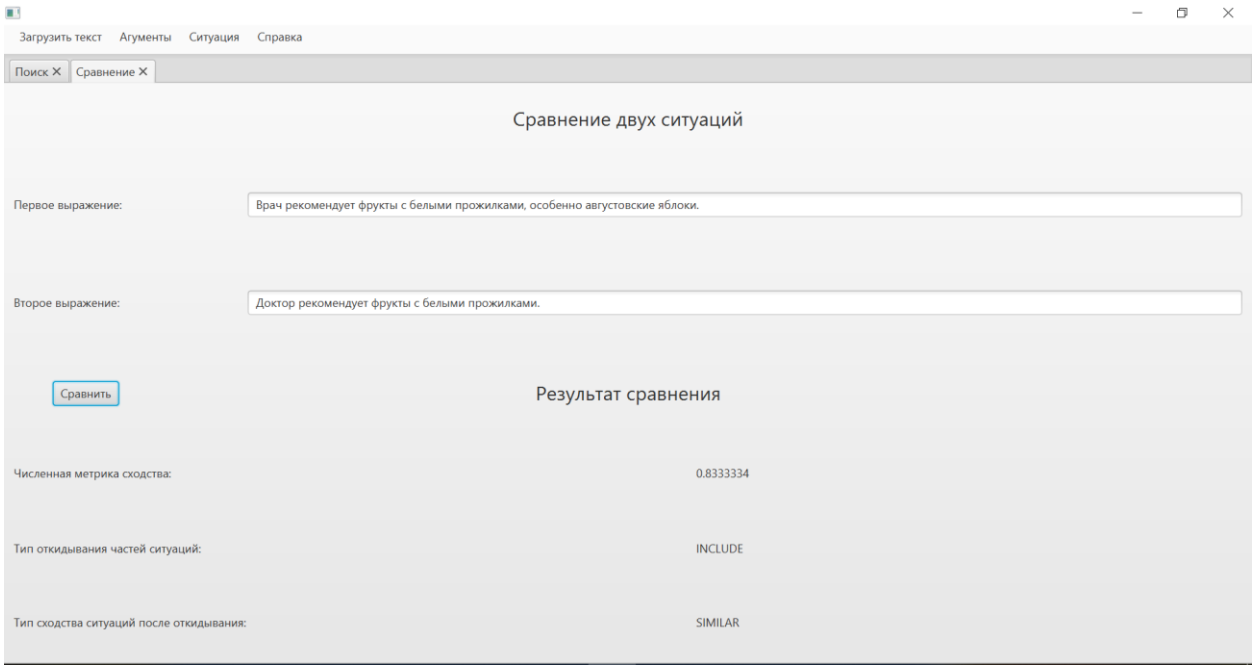


Рисунок 24 — Сравнение ситуаций по утверждениям естественного языка

Краткое описание программы, ссылка на исходный код, контактные данные автора указаны на информационной странице программной системы, изображенной на рисунке 25.

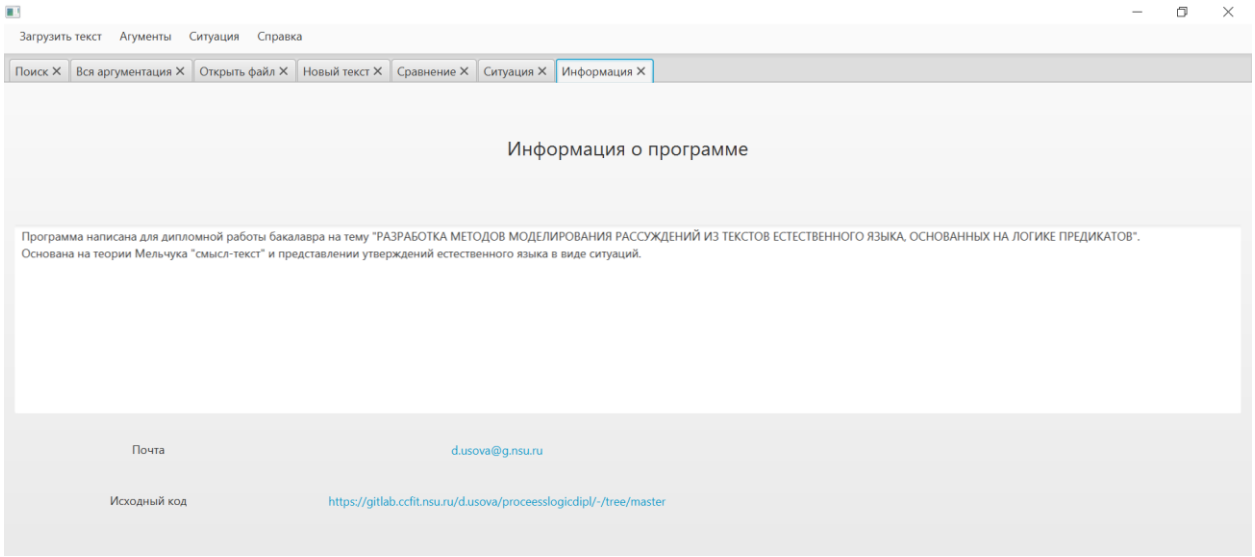


Рисунок 25 — Информационная страница

3.12 Тестирование программной системы

Система была протестирована на предмет корректности реализации описанного функционала. На стороне сервера написаны автоматические тесты для проверки базовых возможностей:

1. Загрузка текста на естественном языке для извлечения аргументов и получение сохраненных аргументов.

Были взяты реальные новостные тексты [15; 16]. Тест проверяет то, сколько и какие аргументы были извлечены в процессе загрузки текста, а также то, что полученные аргументы доступны через метод получения всех сохраненных аргументов.

2. Получение ситуаций для текстов естественного языка

Рассмотрен пример получения ситуации для простого предложения «Врач рекомендует фрукты с белыми прожилками». Сравниваются ожидаемая ситуация и ситуация, полученная системой.

3. Сравнение ситуаций, построенных по двум утверждениям на естественном языке

Для каждого возможного типа отношений между ситуациями написан тест, проверяющий корректность результата сравнения ситуаций. Тестовые данные описаны в таблице 2.

Таблица 2 — Тестовые данные для сравнения ситуаций

Первое утверждение	Второе утверждение	Тип отношений
Врач рекомендует фрукты с белыми прожилками.	Врач рекомендует фрукты с белыми прожилками.	EQUAL, SAME
Врач рекомендует фрукты с белыми прожилками.	Доктор рекомендует фрукты с белыми прожилками.	SIMILAR, SAME
Врач рекомендует фрукты с белыми прожилками.	Эксперт рекомендует фрукты с бесцветными прожилками.	GENERALIZATION, SAME
Врач рекомендует фрукты с белыми прожилками, особенно спелые мандарины.	Врач рекомендует фрукты с белыми прожилками.	EQUAL, INCLUDE
Врач рекомендует фрукты с белыми прожилками, особенно спелые мандарины.	Доктор рекомендует фрукты с белыми прожилками.	SIMILAR, INCLUDE

Врач рекомендует фрукты с белыми прожилками, особенно спелые мандарины.	Эксперт рекомендует фрукты с белыми прожилками.	GENERALIZATION, INCLUDE
Доктор рекомендует фрукты с белыми прожилками, особенно спелые мандарины.	Врач рекомендует фрукты с белыми прожилками, врач любит фрукты.	SIMILAR, INTERSECTION
Эксперт рекомендует фрукты с белыми прожилками, особенно спелые мандарины.	Врач рекомендует фрукты с белыми прожилками, врач любит фрукты.	GENERALIZATION, INTERSECTION
Врач рекомендует фрукты с белыми прожилками, особенно спелые мандарины.	Врач рекомендует фрукты с белыми прожилками, врач любит фрукты.	EQUAL, INTERSECTION

4. Поиск аргументации для утверждения пользователя

Проверялся поиск аргументации для различных утверждений, рассмотрены все варианты типов отношений между ситуацией на основе предложения пользователя и ситуацией в выводе аргумента. В тесте проверяется, содержит ли результат программной системы ожидаемый аргумент с необходимой метрикой сходства. Тестовые данные приведены в таблице 3.

Таблица 3 — Тестовые данные для поиска аргументации

Утверждение	Аргумент	Тип сходства
Бурляева посоветовала есть их с белыми прожилками, врачи совертуют овощи.	В них содержится клетчатка -> Бурляева посоветовала есть их с белыми прожилками.	EQUAL, INCLUDE
Бурляева посоветовала есть их с бесцветными прожилками.	В них содержится клетчатка -> Бурляева посоветовала есть их с белыми прожилками.	EQUAL, SAME
Бурляева посоветовала есть их с бесцветными прожилками, врачи	В них содержится клетчатка -> Бурляева посоветовала есть их с белыми прожилками.	GENERALIZATION, INCLUDE

советуют овощи.		
Бурляева посоветовала есть их с бесцветными прожилками.	В них содержится клетчатка -> Бурляева посоветовала есть их с белыми прожилками	GENERALIZATION, SAME
Бурляева посоветовала есть их с прожилками, врачи советуют фрукты.	В них содержится клетчатка -> Бурляева посоветовала есть их с белыми прожилками.	EQUAL, INTERSECTION
Части аллергических ответов можно будет избежать, врачи советуют фрукты.	Также врач рассказала, что аллергический ответ возникает на масла в кожице мандарина -> если человеку предложить уже очищенный мандарин, части аллергических реакций можно будет избежать.	SIMILAR, INTERSECTION
Если человеку предложить уже очищенный мандарин, части аллергических ответов можно будет избежать.	Также врач рассказала, что аллергический ответ возникает на масла в кожице мандарина -> если человеку предложить уже очищенный мандарин, части аллергических реакций можно будет избежать.	SIMILAR, SAME
Части аллергических ответов можно будет избежать.	Также врач рассказала, что аллергический ответ возникает на масла в кожице мандарина -> если человеку предложить уже очищенный мандарин, части аллергических реакций можно будет избежать.	SIMILAR, INCLUDE
От части аллергических ответов можно будет спастись, врачи советуют фрукты.	Также врач рассказала, что аллергический ответ возникает на масла в кожице мандарина -> если человеку предложить уже очищенный мандарин, части аллергических реакций можно будет избежать.	GENERALIZATION, INTERSECTION
Если человеку предложить уже очищенный мандарин, части аллергических	Также врач рассказала, что аллергический ответ возникает на масла в кожице мандарина -> если	EQUAL, INCLUDE

реакций можно будет избежать, врачи советуют фрукты.	человеку предложить уже очищенный мандарин, части аллергических реакций можно будет избежать.	

Помимо автоматического тестирования, было проведено ручное тестирование с использованием клиентского приложения.

Получена аргументация для ряда утверждения, например для «Части аллергических реакций можно будет избежать». Результаты поиска изображены на рисунке 18 и отсортированы в порядке убывания численного значения сходства.

Были загружены последние новостные тексты [17], результаты изображены на рисунке 21. Также, была протестирована возможность загружать тексты вручную и через текстовый файл. Результаты изображены на рисунках 19 и 20 соответственно. На рисунке 21 изображен результат получения всех сохраненных аргументов.

Был протестирован функционал получения ситуации для ряда утверждений, например для утверждения «Врач рекомендует фрукты с белыми прожилками», результат на рисунке 22. Были сравнены ситуации для ряда поданных в систему утверждений, например для «Врач рекомендует фрукты с белыми прожилками, особенно августовские яблоки» и «Доктор рекомендует фрукты с белыми прожилками», результат на рисунке 24.

ЗАКЛЮЧЕНИЕ

В ходе работы над темой был получен ряд результатов. Изучены существующие методы формализации знаний и аргументации. В том числе был изучен ряд как русскоязычных, так и иностранных источников литературы.

Выбран подход формализации знаний на основе атомарных диаграмм и теории Мельчука «Смысл-текст», так как он позволяет моделировать неполную динамически изменяющуюся информацию.

Описаны понятия ситуаций, аргумента. Приведен алгоритм преобразования утверждений естественного языка к ситуациям. Также, в целях наполнения системы данными, описан метод извлечения аргументов из текстов естественного языка на основе маркеров аргументации.

Разработаны типы отношений сходства между ситуациями и алгоритмы их сравнения. Отношения сходства позволяют находить семантически схожие утверждения.

Также, на базе идеи рассуждения по аналогии разработаны методы получения аргументации для утверждений естественного языка, для чего введены схемы получения аргументов-гипотез.

На основе описанных методов и алгоритмов разработана программная система, реализующая заявленный в работе функционал. Система была протестирована с помощью автоматических тестов и в ручном режиме.

В процессе работы над темой ВКР был опубликован тезис на международной конференции «Мальцевские чтения» 2020 года [19], было проведено выступление с презентацией.

Работа была представлена на Международной научной студенческой конференции в секции «Информационные технологии» в 2021 году [20]. Работа награждена дипломом I степени.

Выпускная квалификационная работа выполнена мной самостоятельно и с соблюдением правил профессиональной этики. Все использованные в работе материалы и заимствованные принципиальные положения (концепции) из опубликованной научной литературы и других источников имеют ссылки на них. Я несу ответственность за приведенные данные и сделанные выводы.

Я ознакомлена с программой государственной итоговой аттестации, согласно которой обнаружение плагиата, фальсификации данных и ложного цитирования является основанием для не допуска к защите выпускной квалификационной работы и выставления оценки «неудовлетворительно».

Усова Дарья Сергеевна _____

ФИО студента *Подпись студента*

« ____ »_____20 __г.

(заполняется от руки)

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Efstathiou V., Hunter A. An algorithm for generating arguments in classical predicate logic // European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty. – Springer, Berlin, Heidelberg, 2009. – С. 119-130.
2. Михайлов И. С., Тайк З. М. Разработка и реализация модификации алгоритма Rete для нечетких экспертных систем // Вестник Московского энергетического института. Вестник МЭИ. – 2015. – №. 6. – С. 114-119.
3. Chernodub A. et al. Targer: Neural argument mining at your fingertips // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations. – 2019. – С. 195-200.
4. Ненашева Е. О., Пальчунов Д. Е. Разработка автоматизированных методов представления знаний о действиях и ситуациях // Вестник НГУ. Серия: Информационные технологии. 2019. Т. 17, № 3. С. 61–72.
5. Ненашева Е. О., Пальчунов Д. Е. Разработка автоматизированных методов преобразования предложений естественного языка в бескванторные формулы логики предикатов // Вестн. НГУ. Серия: Информационные технологии. 2017. Т. 15, № 3. С. 49–63.
6. Кузнецов С. А. Большой толковый словарь русского языка // Рос. акад. наук, Ин-т лингвист. исслед. — СПб.: Норинт, 2004. — 1534 с.
7. Махасоева О. Г., Пальчунов Д. Е. Автоматизированные методы построения атомарной диаграммы модели по тексту естественного языка // Вестник Новосибирского государственного университета. Серия: Информационные технологии. – 2014. – Т. 12. – №. 2. — 73 с.
8. Шустова М. Разработка автоматических методов анализа аргументации, содержащейся в текстах естественного языка // Международная конференция «Знания-Онтологии-Теории» (ЗОНТ-2019). – 2019. – С. 442-444.
9. Крюков К. В. и др. Меры семантической близости в онтологии // Проблемы управления. – 2010. – №. 5. – 14 с.
10. Логика. Теория и практика аргументации : учебник / И. В. Хоменко. — М. : Издательство Юрайт ; ИД Юрайт, 2011. — 314 с.
11. Chernobay Y. Building Wordnet for Russian Language from Ru. Wiktionary //Conference on Artificial Intelligence and Natural Language. – Springer, Cham, 2017. – С. 113-120.

12. Dea C. et al. Javafx 8: Introduction by example. – New York City, USA : Apress, 2014. – 383 с.
13. Drake J. D., Worsley J. C. Practical PostgreSQL. – " O'Reilly Media, Inc.", 2002. – 621 с.
14. Bauer C., King G. Hibernate in action. – Greenwich CT : Manning, 2005. – Т. 1.
15. В Минздраве назвали суточную норму потребления мандаринов [Электронный ресурс] // Lenta.ru: [сайт]. — URL: https://lenta.ru/news/2020/12/05/tangerine/ (дата обращения: 11.05.2021)
16. На любимом курорте россиян — дефицит жилья. Где в Крыму остались дешевые квартиры и как их купить?: Квартира: Дом [Электронный ресурс] // Lenta.ru : [сайт]. — URL: https://lenta.ru/articles/2020/12/02/crimean_estate/ (дата обращения: 11.05.2021)
17. XML рассылка [Электронный ресурс] // MK.RU Новосибирск : [сайт]. — URL: https://www.mk.ru/rss/science/index.xml (дата обращения: 11.05.2021)
18. Зулин Д. К. Разработка системы контекстного поиска и извлечения аргументации из текста на естественном языке // Материалы 59-й Междунар. науч. студ. конф. 12–23 апреля 2021 г. / Новосиб. гос. ун-т. — Новосибирск : ИПЦ НГУ, 2021. С. 139-139.
19. Усова Д.С. Разработка методов моделирования рассуждений из текстов естественного языка, основанных на логике предикатов // Международная конференция МАЛЬЦЕВСКИЕ ЧТЕНИЯ – 2020. С. 96-96.
20. Усова Д.С. Разработка методов моделирования рассуждений из текстов естественного языка, основанных на логике предикатов // Материалы 59-й Междунар. науч. студ. конф. 12–23 апреля 2021 г. / Новосиб. гос. ун-т. — Новосибирск : ИПЦ НГУ, 2021. С. 164 -164.
21. Palchunov D.E. Axiomatization of Classes of Domain Cases Based on FCA // In: 18th Russian Conference, RCAI 2020, Moscow, Russia, October 10– 16, 2020, Lecture Notes in Artificial Intelligence, vol. 12412, Springer, p. 3-14.
22. Пальчунов Д. Е. Применение анализа формальных понятий для разработки теории моделей предметных областей // Готовится к публикации. – 18 с.

ПРИЛОЖЕНИЕ А

СИСТЕМА АВТОМАТИЧЕСКОГО ИЗВЛЕЧЕНИЯ И АНАЛИЗА АРГУМЕНТАЦИИ ИЗ ТЕКСТОВ ЕСТЕСТВЕННОГО ЯЗЫКА

РУКОВОДСТВО ОПЕРАТОРА

Листов 10

Новосибирск 2021

СОДЕРЖАНИЕ

АННОТАЦИЯ

Данный документ является руководством оператора по установке и применению системы автоматического извлечения и обработки аргументации из текстов естественного языка.

Оформление документа выполнено в соответствии с ГОСТ 19.505-79 «ЕСПД. Руководство оператора» и ГОСТ 19.105-78 «Единая система программной документации (ЕСПД). Общие требования к программным документам (с Изменением N 1)».

1 Назначение программы

1.1 Функциональное назначение программы

Программа позволяет: извлекать аргументацию из текстов естественного языка, преобразовать утверждения к специальной форме – ситуациям, сравнивать ситуации, находить аргументацию для утверждений естественного языка. Тексты для извлечения аргументации могут быть предоставлены пользователем или скачаны в автоматическом режиме.

1. 2 Эксплуатационное назначение программы

В первую очередь программа предназначена для научных сотрудников и студентов для использования в научных и учебных целях в исследованиях на тему моделирования аргументации. Также, программа может быть использована в качестве части более крупных программных продуктов. Например, в качестве модуля поддержки принятия решений.

1.3 Состав функций

1. Автоматическое скачивание последних опубликованных новостных статей с определенного сайта;
2. Возможность загрузки текста вручную и с указанием текстового файла;
3. Автоматическое извлечение аргументации из последних опубликованных на определенном сайте текстов;
4. Получение атомарных диаграмм из утверждений естественного языка;
5. Преобразование атомарных диаграмм к ситуациям;
6. Извлечение и сохранение аргументации в виде структур, содержащих ситуации;
7. Сравнение ситуаций;
8. Поиск аргументации с использованием разработанных схем для введенных утверждений на естественном языке;

2 Условия выполнения программы

2.1 Минимальный состав аппаратных средств

A. Память на диске не менее 200 Мб

B. Оперативная память 8 Мб

C. Процессор модели intel core i3 и выше или аналогичные по характеристикам процессоры amd

D. Устройства ввода/вывода: клавиатура, мышь, монитор

2.1 Минимальный состав программных средств

A. Для запуска программной системы рекомендуется одна из следующих операционных систем:
    a. Windows версии не ниже 8
B. Python версии не ниже 3.7
C. Библиотека feedparser
D. Библиотека beautifulsoup4
E. Библиотека wiki-ru-wordnet
F. Java версии не ниже 11
G. СУБД PostgreSQL версии не ниже 13.1
H. Maven версии не ниже 3.5.4 (необходимо, если есть потребность в самостоятельной сборки проекта)

2.3 Требования к оператору

Требуется человек, знакомый с основами работы с персональным компьютером, который должен: владеть базовыми навыками работы с командной строкой, иметь опыт работы со средством сборки maven, иметь опыт запуска java-программ.

3 Выполнение программы

3.1 Загрузка и запуск программы

Перед первым запуском необходимо создать таблицы данных, для этого необходимо запустить файл *create_tables.sql*.

Перед запуском приложения необходимо запустить сервер для взаимодействия со словарем онтологий, для этого необходимо выполнить команду в корневой директории проекта *python wordnet.py*.

Также, перед запуском основного сервера необходимо настроить параметры в файле application.properties.

Чтобы запустить основной сервер системы, необходимо выполнить в консоли команду *java -Dfile.encoding=UTF-8 -jar path_to_dir/server-1.0-SNAPSHOT.jar --spring.config.location=path_to_dir/application.properties*.

Чтобы запустить клиентскую часть программы, необходимо выполнить *java --module-path "path_to_dir\openjfx-11.0.2_windows-x64_bin-sdk/javafx-sdk-11.0.2/lib//" --add-modules javafx.controls,javafx.fxml -Dfile.encoding=UTF-8 -jar client.jar*.

3.2 Выполнение программы

Интерфейс программы содержит меню для просмотра открытых вкладок и переключения между ними. У каждой вкладки есть имя, отображающее суть функционала данной вкладки. Переключение между вкладками подразумевает нажатие на название необходимой вкладки в меню открытых вкладок, после чего содержимое этой вкладки отобразимся в окне под меню вкладок. В этом окне может быть открыта только одна вкладка одновременно. Интерфейс адаптируется под размер окна приложения. При запуске приложения открыта только вкладка «Поиск», она же открыта в окне ниже. Чтобы закрыть вкладку, необходимо нажать кнопку [x] справа от названия вкладки.

Также, над меню вкладок есть меню для открытия новых вкладок и получения информации о программе. Слева направо содержит следующие кнопки: «Загрузить текст», «Аргументы», «Ситуация», «Справка». При нажатии на каждую из этих кнопок раскрывается лист с дополнительными кнопками.

При нажатии на кнопку «Загрузить текст» раскрывается список кнопок, все они открывают вкладки, функционал которых заключается в загрузке нового текста для извлечения аргументации. В этом списке содержатся следующие кнопки: «Открыть файл», данная кнопка открывает вкладку «Открыть файл»; кнопка «Вставить текст»

открывает вкладку «Новый текст», «Скачать тексты» запускает процесс скачивания последних опубликованных на новостном сайте текстов и извлечение из них аргументов.

При нажатии на кнопку «Аргументы» раскрывается список кнопок, все они связаны с функционалом отображения аргументации. В этом списке содержатся следующие кнопки: «Вся аргументация», открывает вкладку «Вся аргументация»; «Поиск», открывает вкладку «Поиск».

При нажатии на кнопку «Ситуация» открывается список кнопок, связанных с функционалом ситуаций. Среди них: «Получение ситуации», открывает вкладку «Ситуация»; «Сравнение ситуаций», открывает вкладку «Сравнение».

При нажатии на кнопку «Справка» открывается список кнопок. Все они открывают окно с текстом, предоставляющим пользователю информацию о программе. Среди них кнопка «О программе».

Опишем подробнее содержимое каждой из вкладок.

Вкладка «Поиск» в верхней часть окна в центре содержит поле для ввода текста. В него пользователь должен ввести утверждение, к которому необходимо система должна подобрать аргументацию. Справа от поля для ввода находится кнопка, при нажатии на которую на сервер системы отправляется запрос на поиск аргументации для веденного утверждения. После получения ответа от сервера появляется таблица с результатами поиска. Таблицу можно прокручивать по горизонтали и вертикали для просмотра. В каждой строке таблицы содержится один аргумент для введенного пользователем утверждения.

Таблица содержит следующие столбцы:

1. столбец «premise», содержит утверждение, являющееся условием аргумента;
2. столбец «result», содержит заключение аргумента, связанное по смыслу с введенным утверждением;
3. столбец «similarity metric», содержит численную меру сходства между утверждением пользователя и выводом аргумента;
4. столбец «same part similarity type», содержит тип отношений между утверждением пользователя и выводом аргумента после откидывания некоторых подситуаций и предикатов;
5. столбец «structural similarity type», содержит тип отношений между утверждением пользователя и выводом аргумента, соответствующий тому, какая часть подситуаций и предикатов будет откинута.

Во вкладке «Открыть файл» можно выбрать текстовый файл в файловой системе компьютера. Содержимое этого файла будет отправлено как новый источник для поиска

аргументации. Система поддерживает только .txt формат. Поле для ввода в верхней средней части окна отображает путь к выбранному на данный момент файлу. Путь к файлу можно ввести вручную. Ниже поля для ввода располагается окно для просмотра файловой системы компьютера, представляет собой список файлов, располагающихся в выбранной директории на данный момент. Первая строка с текстом «…» осуществляет переход к родительской директории. Если из списка выбрать директорию, то список обновится, в нем будет отображено содержимое этой директории. Если выбран файл, то поле для ввода автоматически заполнится абсолютным путем к выбранному файлу. Под окном выбора файла располагается кнопка «Загрузить файл», она отправляет запрос на извлечение аргументов из содержимого файла. Когда сервер обработает запрос, появится оповещение, отображающее число извлеченных из текста аргументов. Извлеченные аргументы сохраняются на сервере.

Вкладка «Вставить текст» содержит поле для ввода текста в центре окна. В него необходимо вставить непосредственно текст, из которого будет извлекаться аргументация. Под окном для ввода текста располагается кнопка «Загрузить текст», она отправляет запрос на извлечение аргументов из текста файла. Когда сервер обработает запрос, появится оповещение, отображающее число извлеченных из текста аргументов. Извлеченные аргументы сохраняются на сервере.

Вкладка «Все аргументы» содержит кнопку «Обновить» для обновления списка аргументации. В центре окна располагается таблица, в которой отображаются все извлеченные из текстов на данный момент аргументы.

Таблица содержит следующие столбцы:

1. столбец «premise», содержит условие аргумента, является утверждением естественного языка;
2. столбец «result», содержит заключение аргумента, является утверждением естественного языка.

Вкладка «Сравнение» содержит два поля ввода для сравниваемых утверждений на естественном языке. При нажатии на кнопку «Сравнить» в левой центральной части окна, отправляется запрос на сравнение на сервер. Сервер преобразует утверждения к ситуациям и сравнит их. Когда закончится сравнение, его результаты будут размещены в следующих полях на вкладке:

1. Численная метрика сходства;
2. Тип откидывания частей ситуаций;
3. Тип сходства ситуаций после откидывания.

Вкладка «Ситуаций» содержит поля для ввода текста, на основе которого строится набор ситуаций. Справа от ввода располагается кнопка для подтверждения ввода, которая отправляет запрос на сервер на получение набора ситуаций. После окончания обработки сервером запроса, в древовидном представлении в центре окна выводится набор полученных ситуаций. Для каждой ситуации рекурсивно указываются пронумерованные подситуации. В листьях древовидного представления находятся предикаты.

3.3 Завершение программы

Программа завершается стандартными способами: нажатием кнопки [X] для графического интерфейса или нажатием сочетания Ctrl+C в консоли для серверной части приложения.

3.4 Сообщения оператору

В программе не предусмотрено никаких специальных сообщений оператору.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Таблица 4 — Лист регистрации изменений в программном документе «Руководство оператора»

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа	Подпись	Дата
Номер изм.	измененных	замененных	новых	аннулированных					

ПРИЛОЖЕНИЕ Б

СИСТЕМА АВТОМАТИЧЕСКОГО ИЗВЛЕЧЕНИЯ И АНАЛИЗА АРГУМЕНТАЦИИ ИЗ ТЕКСТОВ ЕСТЕСТВЕННОГО ЯЗЫКА
ОПИСАНИЕ ПРОГРАММЫ

Листов 25

Новосибирск 2021

СОДЕРЖАНИЕ

АННОТАЦИЯ

Данный документ является описанием системы автоматического извлечения и анализа аргументации из текстов естественно языка. Система состоит из: основного сервера, написанного на языке Java; сервера для взаимодействия со словарем онтологий, написанного на языке python; клиентского приложения, написанного на языке java.

Документ оформлен в соответствии с ГОСТ 19.402-78 «ЕСПД. Описание программы» и ГОСТ 19.105-78 «Единая система программной документации (ЕСПД)».

1 Общие сведения

1.1 Обозначение и наименование программы

Полное название программы: система для автоматического извлечения и анализа аргументации из текстов естественных языков.

Программная система состоит из следующих файлов:

1. Скрипт wordnet.py для взаимодействия со словарем онтологий
2. Файл create_tables.sql для создания таблиц
3. Файл серверной части server-1.0-SNAPSHOT.jar
4. Файл интерфейсной части javafx-module.jar
5. Общий размер директории с файлами: Мб
6. Язык интерфейса и документации: русский

1.2 Программное обеспечение, необходимое для функционирования программы

A. Для запуска программной системы рекомендуется одна из следующих операционных систем:

- Windows версии не ниже 8

B. Python версии не ниже 3.7

C. Библиотека feedparser

D. Библиотека beautifulsoup4

E. Библиотека wiki-ru-wordnet

F. Java версии не ниже 11

G. СУБД PostgreSQL версии не ниже 13.1

H. Maven версии не ниже 3.5.4 (необходимо, если есть потребность в самостоятельной сборки проекта)

1.3 Языки программирования

Система состоит из нескольких частей:

1. Основное приложение, написано на языке java
2. Сервер на python для взаимодействия со словарем онтологий
3. Таблицы СУБД, которые создаются с помощью .sql скрипта

2 Функциональное назначение программы

2.1 Назначение программы

Программа предназначена для извлечения аргументации в текстах естественного языка, поиска аргументации для утверждений пользователя, представления текстов естественного языка в формализованном виде.

2.2 Сведения о функциональных ограничениях на применение

Программа работает с онтологией wiki-ru-wordnet, возможность добавления новых понятий в онтологию возможна только через добавление данных в официальный русский Викисловарь.

Добавление текстов для анализа возможно через указание тестового файла для анализа, добавление текста через графический интерфейс программы, автоматически путем скачиванием последних опубликованных статей с определенного сайта. Для загрузки через файл подходят только .txt файлы в кодировке UTF-8.

Получение атомарных диаграмм предложений выполняется с помощью программы LogicText. Добавление новых данных в словари, которые используют этот модуль, на данный момент невозможно, так как программа не распространяется открыто.

3 Описание логической структуры

3.1 Структура программы

Программа состоит из двух модулей:

a. «server»

b. «javafx-module»

Также, в директории, в которой содержатся данные модели, находится файл wordnet.py, предназначенный для взаимодействия со словарем онтологий wiki-ru-wordnet. Данный скрипт написан на языке python и представляет собой HTTP сервер, принимающий запросы на предоставление информации об онтологических связях слов. Далее будем называть его сервером онтологий. Используются библиотека wiki_ru_wordnet для взаимодействия со словарем онтологий.

3.1.1 Модуль server

Модуль server является основным сервером приложения, реализован на языке java. Использует библиотеку LogicText для получения атомарных диаграмм предложений, фреймворк Spring Boot для организации HTTP сервера. Основным функционалом сервера является извлечение, обработка, поиск, сохранение аргументации, предоставление возможности получения аргументации.

Модуль server содержит основной пакет ru.nsu.usova.dipl, он содержит в себе директории src.main.java, src.main.python, src.main.resources:

Директория src.main.python содержит единственный файл text_extractor.py для скачивания последних опубликованных новостных текстов с сайта mk.ru.

Директория src.main.java содержит:

a. controllers

Предоставляет возможность взаимодействия с сервером с помощью протокола http.

b. logictext

Предоставляет возможность взаимодействия с библиотекой LogicText

c. ontology

Предоставляет возможность взаимодействия с сервером онтологий с помощью HTTP запросов.

d. parser

Предоставляет возможность поиска аргументации в текстах и преобразования аргументов к наборам атомарных диаграмм и ситуациям.

e. scenario

Реализует основные сценарии взаимодействия с системой: извлечение аргументации, поиск аргументации

f. situation

Содержит основные сущности, представляющие ситуацию и связи между ними, взаимодействия с базой данных.

Директория src.main.resources содержит файлы:

a. reasoningTemplates.txt

Содержит JSON структуру, являющуюся списком регулярных выражений для поиска предложений, содержащих маркеры аргументации. Для каждого регулярного выражения указано направление аргументации.

b. signs.txt

Содержит списков знаков препинания.

c. application.properties

Файл для настройки сервера.

Пакет controllers содержит пакет model для сущностей, необходимых для http взаимодействий. Также, пакет controllers содержит класс ArgumentController, в котором реализованы HTTP конечные точки для: получения всех аргументов; извлечения аргументов из текста; получения аргументации для утверждения; получения набора ситуаций для текста; сравнения ситуаций.

Пакет controllers model содержит:

a. Класс LoadTextInfo для передачи информации о количестве извлеченных из текста аргументов;

b. Класс ReasononingRequest для передачи информации о запросе пользователя для извлечения аргументации;

c. Класс CompareRequest для передачи двух сравниваемых утверждений.

Пакет logictext содержит класс LogicTextInteraction для вызова метода библиотеки LogicText для получения атомарных диаграмм текста.

Пакет ontology содержит пакет model, который содержит единственный класс OntologyRelated. Данный класс необходим для передачи и сравнения информации о синонимичных и родительских понятий для определенного слова. Также, пакет ontology содержит класс WordNetUtils, который взаимодействует с сервером онтологий для получения информации о синонимичных и родительских понятиях для заданного слова.

Пакет parser содержит следующие классы:

a. model. DelimInfo

Класс предназначен для передачи информации о направлении аргументации. Направление аргументации определяется порядком условия и заключения в предложении относительно маркера аргументации.

b. ExtractReasoning

Класс предназначен для обнаружения в текстах естественного языка аргументации по маркерам, извлечения атомарных диаграмм из извлеченных аргументов и выполнения подготовительных шагов к переходу к ситуациям.

c. ParserUtils

Содержит вспомогательные методы для парсинга аргументации из текстов.

d. TextExtractor

Класс предназначен для чтения текстов из файлов и разбиения текстов на статьи по заданному разделителю.

Пакет scenario содержит классы:

a. model. LinkMetric

Класс для передачи информации об отношении сходства между ситуацией и заключением аргумента.

b. ArgumentExtractorService

Интерфейс для сервиса, предоставляющего возможность извлекать из текста на естественном языке ситуации, для поиска ближайшей ситуации из базы знаний к заданной, для поиска аргументации для заданной ситуации.

c. ArgumentExtractorServiceBean

Реализация интерфейса ArgumentExtractorService.

d. SituationMining

Содержит методы для инициализации приложения, которые из переданного в параметры программы файла извлекают аргументацию и сохраняют начальное состояние базы знаний.

Пакет situation содержит пакеты model, service и repository, а также следующие классы:

a. DbIterator

Класс-итератор для таблицы базы данных, позволяющий выкачивать элементы таблицы частями фиксированного размера. Это необходимо на случай, если таблица слишком большая и не помещается в оперативную память компьютера.

b. DbComponentFactory

Класс для создания итераторов для таблиц баз данных.

c. DbOperationsService

Интерфейс для сохранения элементов таблиц Situation и SituationLink.

d. DbOperationsServiceBean

Реализация интерфейса DbOperationsService.

e. SituationUtils

Содержит вспомогательные методы для алгоритмов сравнения ситуаций.

Пакет situation.model содержит пакет metric и следующие классы:

a. Situation

Сущность для представления внутренней структуры ситуации. Является сущностью базы данных.

b. ReasoningConstruction

Сущность для представления информации об аргументе. Является промежуточным представлением перед преобразованием посылки и заключения аргумента к ситуациям.

c. SituationLink

Сущность для итогового представления аргумента, в которой посылка и заключение представлены в виде ситуаций. Является сущностью базы данных.

d. SituationQuestions

Двухместный предикат.

Пакет situation.model.metric содержит классы:

a. SamePartRelationType

Возможные типы отношений между ситуациями после откидывания некоторых подситуаций и предикатов при сравнении.

b. StructuralRelationType

Возможные типы откидывания подситуаций и предикатов при сравнении.

c. SituationMetric

Класс для передачи информации об отношении сходства между ситуациями.

Пакет situation.repository содержит CRUD репозитории для взаимодействия с базой данных. Содержит следующие классы:

a. BatchRepository

Интерфейс, позволяющий находить максимальный ключ для таблицы в базе данных и получать элементы таблице по заданному промежутку идентификаторов.

b. SituationLinkRepository

CRUD репозиторий, наследующий интерфейс BatchRepository для таблицы SituationLink.

c. SituationRepository

CRUD репозиторий, наследующий интерфейс BatchRepository для таблицы Situation.

Пакет situation.service содержит классы:

a. DbOperationService

Интерфейс для логики сохранения ситуаций и аргументов

b. DbOperationServiceBean

Реализация DbOperationService.

c. SituationsCompareService

Интерфейс для функционала сравнения ситуаций.

d. SituationsCompareServiceBean

Реализация SituationsCompareService.

3.1.2 Модуль javafx-module

Данный модуль является десктопным приложением на языке java, использующий фреймворк javafx. Модуль предоставляет интерфейсную часть программной системы.

Содержит основную директорию src.main, которая в свою очередь содержит директории java и resources. Директория java содержит пакет ru.nsu.usova.dipl, где содержится основной код приложения. Директория resources содержит fxml файлы с базовой статической версией отдельных частей интерфейса программы. Директория resources содержит следующие файлы:

a. load_single_text.fxml

Соответствует вкладке «Вставить текст». Данная вкладка предназначена для отправки на сервер текста для извлечения аргументации.

b. main.fxml

Соответствует интерфейсу главного меню программы и меню для вкладок.

c. open_file.fxml

Соответствует вкладке «Открыть файл». Данная вкладка предназначена для отображения файловой системы и выбора файла, содержимое которого будет отправлено на сервер для извлечения аргументации.

d. view_all.fxml

Соответствует вкладке «Все аргументы». Вкладка предназначена для просмотра всех имеющихся в базе знаний аргументов.

e. view_args.fxml

Соответствует вкладке «Поиск». Данная вкладка предназначена для отображения аргументации для заданного пользователем утверждения.

f. compare_situations.fxml

Соответствует вкладке «Сравнить». Вкладка предназначена для сравнения ситуаций.

g. info.fxml

Соответствует вкладке «О программе».

h. open_situation.fxml

Соответствует вкладке «Ситуаций». Вкладка предназначена для получения ситуации на основе утверждения пользователя.

Пакет ru.nsu.usova.dipl содержит класс Main, который является точкой входа приложения и запускает начальный вид интерфейса main.fxml. Также, он содержит пакет controller для контроллеров отдельных элементов интерфейса, пакет element для классов с общими вспомогательными методами, пакет model, который содержит сущности для коммуникации с сервером. Эти сущности отправляются в качестве тела запроса в виде JSON структуры или принимаются как ответ от сервера в виде JSON структуры.

Пакет controller содержит следующие классы:

a. AllArgumentsController

Контроллер для view_all.fxml, отправляет на сервер запрос на получение всех аргументов из базы знаний и выводит таблицу с результатами запроса.

b. DownloadTextController

Контроллер для кнопки «Скачать тексты», отправляет запрос на сервер на скачивание последних опубликованных текстов с определенного сайта и извлечение из них аргументов.

c. LoadSingleTextController

Контроллер для load_single_text.fxml, отправляет запрос на извлечение аргументов из введенного пользователем текста.

d. MainViewController

Контроллер для main.fxml, открывает начальные вкладки приложения и предоставляет функции для открытия новых вкладок.

e. OpenFileController

Контроллер для open_file.fxml отображает файловую системы компьютера и предоставляет возможность выбрать файл, содержимое которого отправится с запросом на сервер на извлечение аргументации.

f. SearchController

Контроллер view_args.fxml, отправляет на сервер запрос на поиск аргументации для введенного пользователем утверждения и отображает результаты запроса.

g. SituationsCompareController

Контроллер для compare_situations.fxml. Отправляет на сервер запрос для сравнения ситуаций, построенных по двум введенным утверждениям

h. SituationController

Контроллер для open_situation.fxml. Отправляет запрос на сервер для получения ситуации на основе утверждения пользователя.

Пакет element содержит:

a. ButtonEventHandler

Обработчик кнопки для выбора файла. Используется в OpenFileController.

b. FxElementsUtils

Реализует ряд общих методов для взаимодействия с библиотекой javafx, в том числе: открытие новой вкладки с заданным в .fxml файле содержимым; отображение таблицы; открытие нотификационного сообщения; отображение ситуации.

Пакет model содержит:

a. CompareRequest

Сущность для передачи информации о сравниваемых утверждениях.

b. LoadTextInfo

Сущность для передачи информации об извлеченных аргументах.

c. ReasoningTable

Сущность для передачи информации об аргументе. Части аргумента являются

d. ReasononingRequest

Сущность для передачи текста, из которого извлекаются аргументы.

e. Situation

Сущность для передачи информации о внутреннем содержании ситуации.

f. SituationLink

Сущность для передачи информации об аргументе.

g. SituationMetric

Сущность для отображения отношения сходства между ситуациями.

h. SituationQuestion

Сущность для отображения двухместного предиката.

3.2 Алгоритм программы

Программа предоставляет ряд конечных точек со стороны сервера и ряд страниц со стороны графического приложения для реализации заявленных функциональных возможностей. Опишем используемые в системе алгоритмы.

3.2.1 Получение текстов

Новостные тексты для извлечения аргументов можно получить тремя способами:

1. Указав непосредственно текст новостной статьи в форме для ввода в графическом приложении. Графическое приложение отправит HTTP запрос на основной сервер с текстом в качестве параметра. Для этого существует конечная точка с адресом «/text/load», описание которой дано в последующих главах.
2. Указав в графическом приложении название текстового файла на файловой системе компьютера. Графическое приложение извлечет содержимое текстового файла и отправит HTTP запрос на основной сервер с текстом в качестве параметра. Для этого также используется конечная точка с адресом «/text/load».
3. Запустив через операцию в графическом приложении скачивание текстов. Для этого отправляется HTTP запрос на основной сервер на адрес «text/download/». В рамках данного запроса запускается скрипт, скачивающий по протоколу RSS адреса последних опубликованных новостных статей с новостного сайта. Затем скачивается содержимое страниц по полученным ссылкам, из которых извлекается непосредственно текст новостной статьи.

После получения текста из параметра запроса или путем скачивания система начнет извлечение аргументов из него.

3.2.2 Извлечение аргументов из текста

В настройках дан файл со списком регулярных выражений для определения того, содержит ли предложение маркер аргументации. Для каждого из них обозначено, как посылка и заключение аргумента располагаются относительно маркера: сначала условие, затем вывод; сначала вывод, затем условие. Когда в систему поступает новостной текст, он разбивается на предложения, для каждого проверяется, удовлетворяет ли оно одному из заданных в файле регулярных выражений. Если удовлетворяет, то предложение разбивается на части до и после маркера аргументации. В зависимости от заданного для направления определяется, что является посылкой аргумента, а что заключением. Если маркер аргументации является первым словом в предложении, то заключением аргумента является все предложение после него, посылкой является предшествующее предложение.

Перед сохранением в СУБД посылка и заключение из утверждений естественного языка преобразуются в ситуации.

3.2.3 Преобразование утверждения на естественном языке к ситуациям

Для утверждений естественного языка необходимо получить многоместные предикаты. Новостные статьи обрабатывается с помощью программной системы LogicText. На выходе получается набор многоместных предикатов с указанием того, к какому предложению каждый из них относится и разрешенными местоимениями.

В соответствии с алгоритмом преобразования, описанным в методической части дипломной работы, обязательным шагом является выделение прилагательных в отдельный предикат с глаголом «являться».

Для каждого многоместного предиката необходимо указать новую константу-ситуацию и преобразовать к набору двухместных. Глаголы выделяются в двухместные предикаты с типом «Что делать». Далее, каждое место многоместного предиката выделяется в отдельный двухместный.

Таким образом, получаем набор двухместных предикатов, относящихся к одной подситуации, которые объединятся в единую ситуацию. Полученные данные сохраняются в структуре Situation (таблица 8).

Если из текста извлекли аргумент, то необходимо получить отдельные ситуации для его условия и вывода, затем сохранить полученные данные в структуре SituationLink (таблцица 7).

3.2.4 Сравнение ситуаций

Ситуации в программной реализации состоят из:

1. Набора двухместных предикатов;
2. Набора подситуаций.

При создании ситуации в ней есть или только подситуации, или только двухместные предикаты.

Перечислим варианты сравнения ситуаций:

A. Обе ситуации содержат только подситуации;

B. Обе ситуации содержат только двухместные предикаты;

C. Одна ситуация содержит только подситуации, другая только двухместные предикаты.

Рассмотрим алгоритм варианта A. Подситуации обеих ситуаций нумеруются в натуральном порядке. Для номеров ситуации S_1 с меньшим числом подситуаций

формируется упорядоченное без повторений множество перестановок q1. Для номеров ситуации S_2 с большим либо равным числом подситуаций формируется упорядоченное без повторений множество перестановок q2, размер каждого набора в перестановке при этом равен числу подситуаций S_1.

Метрика сходства для ситуаций SituationMetric (таблица 9) содержит: численную метрику сходства, тип откидывания частей ситуаций structuralRelationType, тип сходства ситуаций после откидывания лишнего samePartRelationType.

Для каждого из наборов в q1 перебирается каждый набор в q2, для которых последовательно сравниваются подситуации с соответствующими номерами из ситуаций S_1 и S_2 соответственно. При сравнении пар подситуаций численные метрики сходства суммируются. После сравнения всех пар подситуаций для конкретных наборов в q1 и q2, вычисляется среднее от численной метрики сходства.

Тип сходства samePartRelationType после откидывания лишнего равен типу с наименьшим приоритетом среди сравниваемых подситуаций. Приоритеты типа: EQUAL = 3, SIMILAR = 2, GENERALIZATION = 1.

Тип откидывания частей ситуаций structuralRelationType равен типу с наименьшим приоритетом из типов сравниваемых подситуаций и типа, соответствующего S_1 и S_2. Тип для S_1 и S_2 зависит от того, отличается ли у них число подситуаций. Если число подситуаций равно, то назначается тип SAME, иначе INCLUDE. Приоритеты типа: SAME = 3, INCLUDE = 2, INTERSECTION = 1.

В процессе сравнения пар наборов из q1 и q2 выбирается максимум по численной метрике сходства между ситуациями.

Рассмотрим алгоритм варианта В. Для сравнения двух наборов предикатов необходимо выбрать пересекающиеся по типам и сравнить их значения.

В зависимости от того, как пересекаются множества типов предикатов двух сравниваемых ситуаций, назначается тип отношения structuralRelationType. Если наборы полностью совпадают по типам, то назначается тип SAME. Если все типы одного набора включены в другой, то назначается тип INCLUDE. Иначе, назначается тип INTERSECTION.

Для каждой пары предикатов, входящих во множество пересекающихся типов, сравниваются значения с помощью запроса к серверу онтологий. Если слова являются одинаковыми, синонимичными или одно обобщает другое, то назначается численное значение схожести, равное 1, и соответствующее значение типа samePartRelationType.

Рассмотрим алгоритм варианта С. Пусть ситуация S_1 содержит только подситуации, S_2 содержит только предикаты. Необходимо сравнить каждую подситуацию S_1 с S_2 и выбрать максимум по численной мере сходства.

3.2.5 Поиск аргументации для утверждения пользователя

Для поиска аргументации для утверждения пользователя необходимо:

1. Построить ситуацию для утверждения пользователя;
2. Сравнить вывод каждого сохраненного аргумента с построенной по утверждению пользователя ситуацией;
3. Добавить в результаты поиска аргументы, численное расстояние до которых выше заданного порога.

3.3 Связи между составными частями программы

Связи между Python сервером wordnet.py и основным сервером, а также основным сервером и интерфейсной частью системы реализованы с помощью HTTP запросов.

3.4 Связи программы с другими программами

Программная система связана с:

1. Интерфейсом командной строки операционной системы;
2. Интерпретатором языка Python, необходимым для сервера онтологий;
3. Виртуальной машиной JVM, необходимой для основного сервера и клиентского приложения.

4 Используемые программные средства

Для корректной работы программной системы необходимо использование следующих минимальных требований к программному и аппаратному обеспечению:

1. Для запуска программной системы рекомендуется одна из следующих операционных систем:
    a. Windows версии не ниже 8
2. Python версии не ниже 3.7
3. Библиотека feedparser
4. Библиотека beautifulsoup4
5. Библиотека wiki-ru-wordnet
6. Java версии не ниже 11
7. СУБД PostgreSQL версии не ниже 13.1
8. Maven версии не ниже 3.5.4 (необходимо, если есть потребность в самостоятельной сборки проекта)

5 Вызов и загрузка

Перед первым запуском необходимо создать таблицы данных, для этого необходимо запустить файл *create_tables.sql*.

Перед запуском приложения необходимо запустить сервер для взаимодействия со словарем онтологий, для этого необходимо выполнить команду в корневой директории проекта *python wordnet.py*.

Также, перед запуском основного сервера необходимо настроить параметры в файле application.properties. Параметры описаны в таблице 6.

Чтобы запустить основной сервер системы, необходимо выполнить в консоли команду *java -Dfile.encoding=UTF-8 -jar path_to_dir/server-1.0-SNAPSHOT.jar --spring.config.location=path_to_dir/application.properties*.

Чтобы запустить клиентскую часть программы, необходимо выполнить *java --module-path "path_to_dir/openjfx-11.0.2_windows-x64_bin-sdk/javafx-sdk-11.0.2/lib/" --add-modules javafx.controls,javafx.fxml -Dfile.encoding=UTF-8 -jar client.jar*.

6 Входные данные

Для интерфейсной части возможен ввод любых символов с клавиатуры и выбор любого .txt файла в кодировке UTF-8 для отправки текста на извлечение аргументов.

Python сервер для взаимодействия со словарем онтологий предоставляет возможность отправлять запросы на следующий адрес «http://url: port/synsets?word=your_word». Через параметр «word» передается слово, для которого необходимо найти синонимичные и общие понятия в словаре.

Для функционирования модуля «server» необходим ряд конфигурационных файлов, они предоставляются вместе с распространяемым архивом. При необходимости их можно редактировать. Список конфигурационных файлов и требования к их содержанию приведены в таблицах 5-6.

Таблица 5 — Файлы для настройки модуля «server»

Название файла	Описание
reasoningTemplates.txt	Содержит JSON, представляющий собой key-value структуру. Ключом является регулярное выражение, позволяющее определить, есть ли в предложении маркер аргументации. Значением является структура, содержащая следующие бинарное поле «direction», обозначающие порядок условия и заключения аргумента относительно ключа пары.
signs.txt	Содержит возможные знаки препинания. На каждой строке файла один знак препинания.
application.properties	Файл для настройки параметров приложения. Используемые поля приведены в таблице 2.

Таблица 6 —Используемые настройки application.properties

Название	Описание	Значение по умолчанию
spring.datasource.url	Адрес базы данных	jdbc:postgresql://localhost:5432/dipl
spring.datasource.username	Логин пользователя базы данных	Postgres

spring.datasource.password	Пароль пользователя базы данных	123
spring.datasource.driver-class-name	Название драйвера базы данных	org.postgresql.Driver
spring.jpa.database	Название используемой базы данных	postgresql
spring.jpa.database-platform	Название SQL диалекта для фреймворка hibernate	org.hibernate.dialect.PostgreSQL10Dialect
extract.script	Абсолютный путь к скрипту для скачивания новостных статей. Скрипт распространяется вместе с основной программой	

Модуль server предоставляет несколько конечных точек для взаимодействия. Перечислим основные конечные точки и входные данные для них в таблице 9.

Прежде чем описывать входные данные отдельно для каждой конечной точки, опишем основные структуры, используемые в качестве входных параметров в таблицах 7 и 8.

Таблица 7 — Структура ReasoningRequest

Название поля	Тип данных	Описание
statement	String	Поле для передачи текстовых данных

Таблица 8 — Структура CompareRequest

Название поля	Тип данных	Описание
firstPhrase	String	Утверждение на естественном языке
secondPhrase	String	Утверждение на естественном языке

Таблица 9 — Входные значения конечных точек основного сервера

URL	Тип запроса	Описание	Тип входных параметров	Описание параметра
/argument/all	GET	Получение всех сохраненных	Нет	

		аргументов.		
/argument/statement	POST	Получить аргументацию для предложения.	ReasoningRequest	Содержит утверждение, для которого производится поиск аргументации. В теле запроса в формате JSON.
/text/load	POST	Загрузка текста для извлечения аргументации.	ReasoningRequest	Содержит текст. В теле запроса в формате JSON.
/text/download	POST	Автоматическое скачивание последних 20 новостных статей с новостного сайта и поиск аргументации в них.	Нет	
/situation	GET	Получение ситуации на основе предложения на естественном языке.	String	Текст для преобразования в утверждение. В теле запроса в формате JSON.
/situation/compare	GET	Преобразование двух утверждений к ситуациям и их сравнение.	CompareRequest	Содержит сравниваемые утверждения. В тезе запроса в формате JSON.

7 Выходные данные

Выходными данными графического приложения являются элементы интерфейса, в том числе текстовые оповещение в ответ на действия пользователя.

На выход python сервера для взаимодействия со словарем онтологий на конечной точке с адресом «/synsets?word=your_word» отдается json структура, описанная в таблице 10.

Таблица 10 — Выходная структура конечной точки сервера взаимодействия с онтологиями

Тип данных	Название	Описание
List<String>	synsets	Список уникальных синонимичных понятий для заданного слова.
List<String>	hyps	Список уникальных родительских понятий для заданного слова.

Прежде чем описывать выходные данные конечных точек основного сервера, опишем основные используемые для этого типы данных в таблицах 11-15.

Таблица 11 — Структура аргумента SituationLink

Тип данных	Название	Описание
Situation	premiseSituation	Условие аргумента в виде ситуации.
Situation	resultSituation	Результат аргумента в виде ситуации.
String	premise	Условие аргумента в виде утверждения естественного языка.
String	result	Результат аргумента в виде утверждения естественного языка.

Таблица 12 — Структура ситуации Situation

Тип данных	Название	Описание
Map<String, String>	Questions	Набор двухместных предикатов.
List<Situation>	childSituations	Подситуации.

Таблица 13 — Структура отношений между двумя ситуациями SituationMetric

Тип данных	Название	Описание
Float	metric	Численная метрика сходства между ситуациями.
String	samePartRelationType	Тип отношений, указывающий на то, как соотносятся ситуаций с учетом откинутых подситуаций и предикатов. Возможные значения: 1. EQUAL – в точности равны 2. SIMILAR – семантически схожи,

		синонимичны 3. GENERAL – одна обобщает другую
String	structuralRelationType	Тип отношений, указывающий на то, какие части ситуаций были откинуты. Возможные значения: 1. SAME – ничего не откинуто в обеих ситуациях; 2. INCLUDE – в одной ситуации откинута часть подситуаций или предикатов, в другой ничего не откинуто; 3. INTERSECTION – в обеих ситуациях было что-то откинуто.

Таблица 14 — Структура отношений между ситуацией в составе аргумента и свободной ситуацией LinkMetric

Тип данных	Название	Описание
Float	metric	Аналогично полю SituationMetric.metric.
String	samePartRelationType	Аналогично полю SituationMetric.samePartRelationType.
String	structuralRelationType	Аналогично полю SituationMetric.structuralRelationType.
SituationLink	link	Аргумент, с заключением которого сравнивается некоторая ситуация.

Таблица 15 — Структура LoadTextInfo

Тип данных	Название	Описание
Integer	extractedArguments	Число извлеченных из текста аргументов.

В таблице 16 представлены конечные точки основного сервера и их выходные данные.

Таблица 16 — Выходные данные конечных точек основного сервера

URL	Тип запроса	Выходные данные	Описание параметра
/argument/all	GET	List<SituationLink>	Список сохраненных на данный момент аргументов в СУБД. Ответ в формате JSON.

/argument/statement	POST	List<LinkMetric>	Список аргументов с указанием метрик сходства заключения и ситуации на основе утверждения пользователя. Ответ в формате JSON.
/text/load	POST	LoadTextInfo	Содержит число извлеченных из текста аргументов. Ответ в формате JSON.
/text/download	POST	LoadTextInfo	Содержит число извлеченных из текстов аргументов. Ответ в формате JSON.
/situation	GET	Situation	Ситуация, составленная по утверждению пользователя. Ответ в формате JSON.
/situation/compare	GET	SituationMetric	Отношение сходства между двумя ситуациями. Ответ в формате JSON.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Таблица 17 —Лист регистрации изменений в программном документе «Описание программы»

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа	Подпись	Дата
Номер изм.	измененных	замененных	новых	аннулированных					