Санкт-Петербургский государственный университет

ЗАХАРОВА Алина Андреевна

Выпускная квалификационная работа

Разрешение стрелочной омонимии в конструкциях с сирконстантами средствами онтологической семантики

Уровень образования: бакалавр
Направление 45.03.02. «*Лингвистика*»
Основная образовательная программа СВ.5106 «*Прикладная, компьютерная и математическая лингвистика (английский язык)*»
Профиль «*Прикладная, компьютерная и математическая лингвистика (английский язык)*»

Научный руководитель:
к.ф.н., ст. преподаватель Добров А.В.

Рецензент:
доцент Митрофанова О. А.

Санкт-Петербург
2020

Оглавление

Введение:

Синтаксический анализ является важным этапом лингвистического анализа текста, так как именно на данном шаге осуществляется разбор структуры предложения. Вместе с тем, в некоторых случаях можно получить несколько вариантов структуры для одного предложения. Такое явления называется синтаксической неоднозначностью. Одна из основных проблем, связанных с этим явлением — проблема комбинаторного взрыва, суть которого в том, что количество версий возрастает в экспоненциальной зависимости от размера анализируемого текста, вследствие чего машинных

ресурсов не хватает для построения и хранения всех версий. Кроме того, подобным образом может быть разобрано не одно и не два предложения, а значительно больше, и тогда производительность парсера снизится, что непосредственно отразится на общем результате обработки текстовых данных.

Если задача требует учета всех корректных версий синтаксического анализа, то необходимо исследовать способы разрешения неоднозначности путем устранения некорректных версий. В данной работе исследуется один из наиболее распространенных видов синтаксической неоднозначности — стрелочная омонимия в конструкциях с сирконстантами, которая разрешается средствами онтологической семантики на основе универсального лингвистического процессора AIIRE (Artificial Intelligence Information Retrieval Engine).

Актуальность темы исследования обусловлена особым интересом исследователей к проблеме синтаксической неоднозначности и методам ее разрешения. Данная задача может решаться не только методами машинного обучения (статистическими или основанными на нейронных сетях, такими как, например, Syntaxnet или Gate, использующие как раз нейронные сети). Такие методы не предполагают участия лингвиста в определении правил, которыми руководствуется система при разрешении неоднозначности, и потому не позволяют ему корректировать их. С другой стороны, можно применять методы компьютерной лингвистики, которые предполагают наличие семантического словаря, онтологии, базы знаний или какого-либо иного лингвистического обеспечения. Они используются семантическим компонентом системы при семантическом анализе и, в частности, обеспечивают выбор семантически допустимых версий синтаксического анализа. Методы компьютерной лингвистики в настоящей задаче востребованы в связи с необходимостью учета всех корректных версий синтаксического анализа в ряде задач автоматического понимания текстов

(Natural Language Understanding). К числу этих задач относятся многовариантный машинный перевод, семантический поиск, извлечение фактической информации (fact extraction) и мнений (opinion mining), а также в области синтаксиса как такового и даже в некоторых задачах психолингвистики. Кроме того, задача разрешения синтаксической неоднозначности до сих пор решена лишь частично.

Целью данной работы является определение возможностей онтологической семантики в разрешении стрелочной омонимии в конструкциях с сирконстантами путем экспериментального исследования на материале синтаксически размеченного корпуса текстов на русском языке; оценка трудоемкости и эффективности данного метода. Для достижения данной цели решаются следующие **задачи**:

1. Создание на основе корпуса репрезентативной выборки употреблений русскоязычных конструкций с сирконстантами, характеризующихся стрелочной омонимией, необходимого и достаточного по содержанию и объему для исследования методов и оценки качества их работы.
   1.1. Выделение типов и подтипов неоднозначных конструкций с сирконстантами, составление их структурных схем и формулирование на основе схем поисковых запросов к корпусу;
   1.2. Разработка средств автоматической выгрузки синтаксически неоднозначных конструкций со стрелочной омонимией из синтаксического подкорпуса Национального Корпуса Русского Языка (далее — НКРЯ), как такого корпуса с синтаксической разметкой, где можно учитывать синтаксические связи и порядок слов при составлении поискового запроса и таким образом получать только те результаты, которые соответствуют цели поиска;
   1.3. Обеспечение необходимого и достаточного объема и содержания выборки для исследования методов разрешения стрелочной

омонимии в конструкциях с сирконстантами средствами онтологической семантики; оценки качества работы таких методов: составление частотных словарей лемм каждой части речи из конструкций по каждому запросу, составление выборки конструкций всех типов, содержащих наиболее частотные леммы, необходимой и достаточной по объему для обеспечения статистической достоверности выводов об исследуемых показателях эффективности автоматического разрешения неоднозначности.

2. Загрузка созданного репрезентативного корпуса конструкций в корпус-менеджер, обеспечивающий возможность автоматической синтаксической и семантической разметки корпуса при помощи лингвистического процессора, выполняющего разрешение синтаксической неоднозначности средствами онтологической семантики.
3. Моделирование понятий, соответствующих значениям лексических единиц, употребляемых в корпусе, в онтологии, с учетом их семантических валентностей, ограничивающих возможности синтаксической интерпретации неоднозначных конструкций; обеспечение корректности версий автоматической синтаксической разметки конструкций лингвистическим процессором путем корректировки и задания семантических отношений на базовых классах концептов онтологии.
4. Анализ и оценка полученных результатов.

Решение данных задач основано на универсальном лингвистическом процессоре AIIRE, встроенной в него онтологии и инструментах ее редактирования. На основе полученных результатов определяется, в какой мере может быть разрешена неоднозначность данного вида с помощью имеющихся средств онтологической семантики. **Объектом** исследования являются возможности автоматического разрешения синтаксической неоднозначности в конструкциях со стрелочной омонимией, обусловленной

факультативностью сирконстантов, средствами онтологической семантики. **Предметом** исследования являются характеристики эффективности методов разрешения стрелочной омонимии средствами онтологической семантики в конструкциях различных типов со стрелочной омонимией, обусловленной факультативностью сирконстантов.

Научная новизна полученных результатов заключается в том, что эффективность данного метода разрешения синтаксической неоднозначности средствами онтологической семантики впервые исследуется на репрезентативном корпусе конструкций со стрелочной омонимией, и экспериментально доказывается не только его эффективность, но и то, что в большинстве случаев для корректного автоматического разрешения неоднозначности может быть достаточно привязки концептов, стоящих за значениями лексических единиц, к корректным базовым классам концептов онтологии, и не требуется дополнительная корректировка онтологических отношений, регулирующих семантические валентности.

Глава 1. СИНТАКСИЧЕСКАЯ НЕОДНОЗНАЧНОСТЬ В КОНСТРУКЦИЯХ С СИРКОНСТАНТАМИ

В данной главе будет дан анализ существующих подходов к синтаксическому анализу предложения; также будут рассмотрены основные понятия, необходимые для решения практической задачи: синтаксический анализ, синтаксическая неоднозначность и причины ее возникновения, омонимичные конструкции с сирконстантами, понятие сирконстантов и их роль в предложении, онтологическая семантика, понятие онтологии, понятие

семантического словаря; применение закона Парето в компьютерном моделировании значений лексических единиц в корпусе текстов.

1.1 Подходы к синтаксическому компьютерному анализу предложения

Компьютерный анализ структуры предложения выполняется в терминах грамматики непосредственных составляющих, в терминах грамматики зависимостей или с использованием комбинированного подхода. Суть грамматики непосредственных составляющих заключается в членении предложения на непосредственные составляющие, например, на именную и глагольную группы. Далее каждая из этих составляющих делится еще части и так до тех пор, пока не будут получены неделимые части предложения. Данный подход в основном ориентирован на языки со строгим порядком слов (например, английский или немецкий), так как в рамках данного подхода структурный порядок слов в синтаксическом дереве и линейный порядок слов в предложении должны совпадать, поскольку составляющие должны быть непрерывны, а в языках со свободным порядком слов такое условие часто не выполняется. При этом элементы предложения не делятся на главные и зависимые. В грамматике зависимостей — наоборот, между частями предложений устанавливают зависимости, т.е. существует «управляющий элемент» [Теньер 1988: 25] и зависящие от него слова. При этом каждый управляющий элемент может иметь несколько зависимых, тогда он образует «пучок» [Теньер 1988: 25], состоящий, соответственно, из главного слова и его зависимых. Главное слово, например, глагол или существительное, а зависимые актанты, которые выражаются существительн
ыми или их эквивалентами, и сирконстанты, которые выражаются наречиями или их эквивалентами. Существенным недостатком такого подхода является то, что в данном подходе все связи между словами трактуются как подчинительные; сочинительные связи указываются в виде особых функций. Кроме того, Теньер, согласно учебному пособию Алпатова В.М. [Алпатов 1999:182], не различал семантические актанты (участников действия) и

синтаксические актанты (соответствующие им члены предложения) Например, в предложении «*Маша играет с Дашей в куклы*» синтаксически выделяется только одно подлежащее «*Маша*», а актантов два «*Маша*» и «*Даша*». Наконец, в грамматике зависимостей синтаксические связи устанавливаются только между словами, а не между группами слов, как в грамматике непосредственных составляющих; т.е. некоторая словоформа X может быть подчинена одной и только одной словоформе У и никак не может быть подчинена целому предложению [Добров 2016:40]. Например, в предложении из работы А.В. Гладкого «*По графику мы работаем в среду*» [Гладкий 1985: 119] возможны две трактовки «В среду мы будем работать в соответствии с графиком, а в остальные дни, возможно, не в соответствии» и «В соответствии с графиком мы должны работать именно в среду». В первом случае по графику относится только к глаголу «*работаем*», а во втором случае — к целому предложению [Добров 2016: 40]. Второй вариант в рамках грамматики зависимости невозможен. Следовательно, данная особенность грамматики зависимостей приводит к тому, что при разборе синтаксически неоднозначных предложений приходится «предполагать неоднозначность самих синтаксических связей» [Добров 2016:40], при этом предполагать их приходится всегда и это, в свою очередь, приводит к тому, что у предложений появляются противоестественные трактовки. Например, в предложении «*Мы едем в Москву*» предложная группа «*в Москву*» относится к глаголу «*едем*», но, если допустить наличие неоднозначности, получится трактовка «Едем мы в направлении Москвы, а все остальные перемещения, возможно, осуществляем в другом направлении» [Добров 2016:40].

Третий (комбинированный) подход заключается в объединении двух первых. Его суть заключается в том, что создается одна структура, в которой отражается и структура зависимостей, и структура составляющих.

Принимая во внимание все вышесказанное, можно прийти к выводу, что наиболее продуктивным для решения задачи снятия синтаксической

омонимии является третий подход, который и был применен в настоящей работе.

1.2 Неоднозначные синтаксические конструкции

Синтаксически неоднозначные конструкции часто встречаются в текстах русском языке. В работе Е.В. Шкурко о них сказано следующее: «Синтаксические омонимы достаточно широко представлены в современном русском языке: они довольно часто встречаются в газетных и журнальных публикациях, в художественных произведениях, в названиях различных учреждений, в речи дикторов радио и телевидения. Однако, несмотря на столь широкое распространение данного явления в русском языке, степень разработанности проблемы синтаксической омонимии все еще недостаточно высока» [Шкурко 2007]. Вместе с тем, в связи с развитием таких направлений, как психолингвистика, актуальность вопроса о снятии синтаксической неоднозначности неуклонно растет. Например, в диссертации Д.А.Черновой используются «психолингвистические механизмы, действующие при синтаксическом анализе неоднозначных предложений в процессе восприятии речи на русском языке» [Чернова 2016]. Наряду с этим, в последние десятилетия в области самого синтаксиса рассматриваются различные методы снятия синтаксической неоднозначности [Федорова 2005].

Как отмечает О.В. Митренина, «Неоднозначные структуры возникают из-за того, что не все связи в предложении устанавливаются однозначно. Связи, неоднозначно определяемые для конкурирующих структур, называют омонимичными связями» [Митренина 2005].

Синтаксический Анализ — «процесс выявления иерархической структуры подчинительных связей предложения естественного языка, в частности русского, в соответствии с грамматикой языка. Результатом

синтаксического анализа является дерево подчинительных связей (дерево синтаксического разбора)» [Окатьев, Гергель 2008]. Н.Н. Леонтьева не приводит отдельного определения для термина «СинтАн» (синтаксический анализ), но фактически определяет его при описании синтаксического компонента системы АПТ (автоматического понимания текстов): «СинтАн обеспечивает в системах АПТ три основных типа информации о структуре предложения … / 1.линейные отношения …; /2.группировку грамматических элементов … ; / 3.отношения зависимости (доминации, иерархические), когда главный член группы определяет форму своих зависимых» [Леонтьева 2006].

Синтаксические структуры, создаваемые в процессе СинтАн, представляют собой объект обработки при семантическом анализе. Например, в онтологии лингвистического процессора AIIRE, на базе которой проводилось настоящее исследование, синтаксический анализ структуры предложений проводитсяпараллельно с семантическим, т.к. сам анализ основ ан на методе межуровневого взаимодействия, о котором более подробно будет сказано ниже.

Автоматический семантический анализ текста по определению О.В. Митрениной «представляет собой процедуру построения семантического представления возможного содержания текста путем обработки обнаруженных в этом тексте синтаксических структур» [Митренина 2005:54]. Таким образом, «Определить синтаксическую структуру предложения — это значит разделить предложение на составляющие части и определить отношения между этими частями» [Митренина 2005:54]. В качестве минимальной составляющей предложения в данном исследовании рассматриваются словоформы.

Грамматические отношения, возникающие между элементами предложения, обозначаются дугами, соединяющими главное слово (хозяин) и зависимое слово (слуга).

К тем же «слугам» можно отнести разных «хозяев». Например, в конструкции «*Детективы схватили Гулю в постели*» Предложная конструкция «в постели» может присоединяться как к глаголу «*схватить*», так и к существительному «*Гуля*». Тогда есть два варианта синтаксического анализа. В первом варианте трактовка будет следующая: детективы поймали девушку по имени Гуля в ее постели, а во втором варианте: детективы поймали Гулю, одетую в постель. Связи, которые неоднозначно определены для конкурирующих структур, будут называться омонимичными, и далее мы рассмотрим так называемую стрелочную омонимию.

1.3 Омонимичные конструкции с сирконстантами

1.3.1 Стрелочная омонимия и ее типы

Стрелочная омонимия подробно описана в работе А.В. Гладкого [Гладкий 1985: 113]. По определению А.В. Гладкого, «самый обычный случай омонимии этого типа состоит в том, что некоторая НС (непосредственная составляющая) при переходе из одной структуры к другой «перевешивается», т.е. меняет хозяина» [Гладкий 1985: 113].

Наиболее частые типы перевешиваемых НС по Гладкому представлены ниже:

1. «предложная или однородная группа, состоящая из предложных групп. При этом чаще всего конкурируют глагол и существительное или два существительных (*чтобы управлять разнообразными химическими реакциями в клетках*). Возможны и другие пары конкурирующих хозяев: причастие-существительное, причастие-прилагательное, деепричастие-существительное» [Гладкий 1985: 113].

2. «наречие или однородная группа, состоящая из наречий. Конкурирующие хозяева те же (*Девочка вытерла тщательно вымытую посуду*)» [там же].

3.«деепричастие или деепричастный оборот (*Директор распорядился продолжать работу, не обращая внимания на протесты*)» [там же].

4.«прилагательное, причастие или группа прилагательного и причастия (*Встретил приятеля его соседа еще трезвого*)» [там же].

5.«существительное в родительном падеже или группа такого существительного в постпозиции к хозяевам (*представлены исследования по истории Института Общих Исследований*)» [там же].

6. «существительное или его группа в другом косвенном падеже (*Перед сваркой плавлением подготавливают поверхности*)» [там же].

7 «придаточное предложение или корень придаточного предложения (*Пришел руководитель аспиранта, о котором я вам говорил*)» [там же].

8.«существительное или группа существительного, вводимая союзом как (*знания общества как системы*)» [там же].

В рамках данного исследования были использованы типы стрелочной омонимии с сирконстантами. т.е. типы под номерами 1-3.

1.3.2 Сирконстанты в синтаксических конструкциях со стрелочной омонимией

По определению Л. Теньера, «сирконстанты выражают обстоятельства (времени, места, способа и пр.), в которых развертывается процесс. Так, в предложении *Alfred fourre toujours son nez partout* «Альфред всегда всюду сует свой нос» имеется два сирконстанта: один — времени (*toujours* 'всегда') и один — места (*partout* «всюду»)» [Теньер 1988:117]. Таким образом, «Функцию сирконстанта всегда берет на себя слово, относящееся к категории наречия, или группа слов, эквивалентная наречию, например, предложная группа с существительным» [Теньер 1988:117-118]. Следовательно, «... существует столько видов сирконстантов, сколько имеется видов наречий:

времени, места, способа действия и т.д.» [Теньер 1988:118]. Количество сирконстантов не так определенно, как количество актантов. Предложение может не иметь ни одного сирконстанта, а может иметь их в неограниченном количестве.

1.3.3 Роль сирконстантов в неоднозначных синтаксических конструкциях

Как отмечает Теньер, «глагольный узел является центром предложения в большинстве европейских языков. Глагол выражает процесс. Так, в предложении *Alfred frappe Bernard* «Альфред ударяет Бернара» процесс выражен глаголом *frappe* «ударяет». Далее идут актанты — живые существа или предметы, которые участвуют в процессе в любом качестве и сирконстанты в роли обстоятельств. Актанты и сирконстанты зависят от глагола и заполняют его синтаксическую валентность» [Теньер 1988:117].

По определению, данному в словаре В.Н.Ярцевой, «валентность — способность глагола вступать в синтаксические связи с другими элементами. У валентности есть ряд важных характеристик, определяющих ее реализацию» [Ярцева (ред.) 1990]. В.Н. Ярцева выделяет следующие характеристики валентности:

1. «Общий тип валентности: активная валентность (способность слова присоединять зависимый элемент) /пассивная валентность (способность с лов присоединяться к господствующему компоненту сочетания)» [Ярцева (ред.) 1990];

2. «Облигаторность валентности: обязательная/факультативная валентность (понятие, соотносимое с сильным и слабым управлением). Слово открывает в предложении ряд позиций, из которых одни заполняются обязательно, другие — нет. Во фразе «*Петр взял книгу из шкафа*» «*книгу*» — обязательная валентность, «*из шкафа*» — факультативная. Обязательной активной валентностью обладают глаголы неполной

предикации («*иметь*», «*ставить*», «*давать*», «*делать*», «*держать*», «*находиться*» и др.) и их узкие синонимы («*представить*», «*оказать*», «*осуществить*» и др.). Среди существительных обязательную валентность имеют имена действия («*приезд отца*»), качества («*красота пейзажа*»), относительные («*отец Марии*»), категориальные («*тип*», «*пример*», «*результат*»), параметрические («*происхождение языка*», «*высота дома*», «*цвет платья*») и др. Отсутствие зависимого компонента может свидетельствовать об изменении значения слова: расширении («*любить красоту*»), сужении [«*пришел отец*» (данной семьи)] или переносе («*взять высоту*» — «*гору*»). Валентность может преобразовываться также в определенных условиях контекста: например, слово «начало» может утрачивать обязательную объектную валентность в условиях анафоры («Прочитать рассказ от начала до конца»), а слово «глаз» получает обязательную определительную валентность во фразе «У нее голубые глаза» [Ярцева (ред.) 1990];

3. «Число валентностей, например одно-, двух-, трехвалентные глаголы» [Ярцева (ред.) 1990];

4. «Синтаксическая функция дополняющего члена: например, при глаголе может быть обязательной валентностью субъектная («*Петр спит*»), объектная («*Он держит ручку*»), обстоятельственная («*Он проживает в Москве*»), предикативная («*Он стал врачом*»)» [Ярцева (ред.) 1990];

5. «Форма дополняющего члена (часть речи, слово или предложение, форма связи), ср.: «*Я знаю это*», «*Я знаю этого человека*» и «*Я знаю, что он пришел*»; «*Он показал мне свой дом*» и «*Он показал на дом*»» [Ярцева (ред.) 1990];

6. «Категориальная семантика слова, реализующего валентность (для глаголов, например, важны такие семантические категории субъекта и объекта, как одушевленность/неодушевленность, конкретность/

абстрактность, исчисляемость/неисчисляемость и др.)» [Ярцева (ред.) 1990].

Возвращаясь ко второй характеристике, можно заметить, что сирконстанты могут занимать как обязательные позиции в предложении, так и необязательные. Следовательно, сирконстанты могут быть факультативными, а факультативность, в свою очередь, может привести к стрелочной омонимии (см. параграф 1.2.3), т.к., если элемент является необязательным, он может либо не использоваться вовсе, либо может быть присоединен к любому из возможных хозяев.

1.4 Онтологическая семантика в контексте разрешения синтаксической неоднозначности

В данной работе будет использован метод снятия синтаксической неоднозначности с помощью средств онтологической семантики.

По определению С. Ниренберга, «онтологическая семантика – это теория значения в естественном языке и подходов к анализу естественного языка, которая использует конструируемую модель мира или онтологию как основной ресурс для извлечения и представления смысла текстов, построения логического вывода для знаний, извлеченных из текстов, а также для генерации текстов исходя из представления их смысла» [Nirenburg 2004: 6].

Иными словами, предполагается использование онтологии для моделирования в ней понятий, соответствующих лексическим единицам, их значений и валентностей. За счет того, что в онтологии выводятся валентности понятий, достигается возможность ограничить количество комбинаторных вариантов трактовки связей внутри предложения или словосочетания. В частности, таким образом можно прописать факультативные валентности, то есть определить, к какому хозяину действительно относится сирконстант и, следовательно, разрешить

неоднозначность, вызванную факультативностью сирконстанта, при условии, что неоднозначность является разрешимой.

Подробнее о реализации метода будет написано во второй главе в разделе 2.2 Разрешение синтаксической неоднозначности средствами онтологической семантики.

1.4.1 Семантические словари для автоматического семантического анализа

В этом разделе будет дан краткий обзор словарей для автоматического семантического анализа. Далее в параграфе 2.1 Виды методов разрешения синтаксической неоднозначности они будут сравниваться с онтологией.

ТКС — это толково-комбинаторный словарь И.А. Мельчука, А.К. Жолковского и Ю.Д. Апресяна [Мельчук 1984], где валентности глагола задаются в виде обобщенной таблицы. ТКС является основным компонентом модели теории «Смысл-Текст», это словарь «нового типа» (Мельчук), который ориентирован на то, чтобы «помочь построить текст», т.е. дать возможность «найти все мыслимые средства выражения идеи», которые подходят к конкретному контексту и «указать правильный способ комбинирования средств во фразе». Словарь также отличает «формализованн ость», т.е. все примеры употребления приводятся в явной форме [Мельчук 1984].

В словаре Н.Н. Леонтьевой [Леонтьева 2006:153] валентности задаются списками лексем для каждого значения каждого слова. Это семантический словарь РУСЛАН (РУсский СЛоварь для АНализа). «Основной направленностью словаря остается ориентация на семантический анализ в масштабе текста, а не только предложения» [там же]. В словаре В.А. Тузова валентности задаются списками рубрик, хотя он их и называл классами. Например, в нем словаре выделяется рубрика «вода», к которой относятся

соответственно понятия «полноводье», «гидрофилия», «водопроницаемость», «водопользование» и др.

1.4.2 Компьютерные лингвистические онтологии

Однозначного определения термина онтологии нет.

Одно из первых описаний онтологии можно найти в работе Р. Нечеса, Р.Е. Файкса, Т. Финина, Т. Грубера и др. [Neches at al. 1991:16-36]. Один из соавторов вышеуказанной работы Т.Р. Грубер в своих дальнейших публикациях приводит определения термина «онтология». Онтология — это «спецификация концептуализации для некоторой общеиспользуемой области дискурса определения классов, отношений, функций и иных объектов» [Gruber 1993:2]. Иными словами, онтология — это формальное описание некоторой области знания, которая представляется в виде структуры, состоящей из объектов, отношений между объектами и атрибутов.

В данной работе используется один из видов онтологии — лингвистическая онтология, которая используется для обработки языковых данных и понимается как множество понятий, которые стоят за лексическими значениями и отношений между ними.

По определению А.В. Доброва «Онтологии состоят из концептов и отношений между ними. Концепты представляют собой формальные (математические, компьютерные) модели понятий, стоящих за значениями лексических единиц» [Добров 2014], которые включают в себя «атрибуты» и связаны между собой отношениями. Последние, по определению А.В. Доброва, представляют собой «класс содержательных связей между концептами двух классов. Все отношения сами по себе являются понятиями и, как и все понятия, могут обозначаться конкретными выражениями (предлогами, союзами, глаголами, сложными наименованиями отношений)»

[Добров 2014]. Концепты, участвующие в отношении, называются субъектом и объектом отношения. Ниже приведены примеры отношений из Конвенций и методики работы с онтологией AIIRE:

«Обратные отношения — это такие отношения A и B, что субъект отношения A является объектом отношения B, а объект отношения A является субъектом отношения B. Пример обратных отношений:

осуществлять действие — r→ обладать объектом действия — o→ объект

Симметричные отношения — это отношения, которые являются обратными сами по отношению к себе, например отношение синонимии.

Ряд важнейших отношений между концептами включает в себя синонимию, гипо-гиперонимию, меро-холонимию. Концепты, связанные с данным концептом через эти отношения, называются синонимами, гипонимами/гиперонимами, меронимами/холонимами.

Отношение «включение элемента» означает, что концепт X класса «совокупность» включает в себя некоторый элемент Y.

Генитивные отношения соответствуют генитивной конструкции в русском языке («*город России*«) и синонимичной ей атрибутивной конструкции («*российский город*«) и указываются у базовых классов существительных (к нижестоящим классам такие отношения переходят по наследованию). Для каждого базового класса устанавливается совокупность отношений с другими базовыми классами [Электронный ресурс] Онтология AIIRE Конвенции и методика по работе с онтологией //URL:http://ontology.aiire.org/static/conventions.html.(дата обращения: 30.08.19)

Предложные отношения являются отношениями либо между значением глагола и значением существительного (приглагольные предлоги,

ср. сидеть в мешке), либо между значениями двух существительных (приименные предлоги, ср. кот в мешке). [там же].

Отношение «включение объектов класса» означает, что концепт X включает в себя (или способен включать) концепты класса Y» [там же].

Совокупность всех отношений, в которых участвует понятие является атрибутом класса или же его сигнификатом, а совокупность экземпляров – денотатом [там же].

По мнению А.В. Доброва, «Онтологии, используемые для а.о.т (автоматической обработки тестов), должны включать в себя отношения, позволяющие производить семантический анализ текста и осуществлять разрешение лексической и синтаксической неоднозначности» [Добров 2014].

В данной работе снятие синтаксической неоднозначности проводилось на основе универсального лингвистического процессора AIIRE и встроенной в него онтологии, более подробное описание которых будет дано в разделе 2.1 Виды методов разрешения синтаксической неоднозначности.

1.5 Закон Парето в компьютерном моделировании значений лексических единиц в корпусе текстов

Основная идея закона Парето заключается в том, что 20% усилий дают 80% результатов, а остальные 80% усилий затрачиваются на достижение последних 20% от результата [Электронный ресурс] Закон Парето //URL:https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%BE%D0%BD_%D0%9F%D0%B0%D1%80%D0%B5%D1%82%D0%BE (дата обращения: 16.10.2019). Иными словами, необходимо определить минимальное количество важных необходимых действий для того, чтобы выполнить бóльшую часть поставленной задачи, затратив при этом меньшее количество усилий и ресурсов. Данный закон часто используется в исследованиях, основанных на корпусах текстов, и может быть использован в

рамках настоящего экспериментального исследования возможностей онтологической семантики в разрешении стрелочной омонимии следующим образом: после получения частотного словаря всех лексических единиц корпуса необходимо определить такую совокупность наиболее частотных лексических единиц корпуса, что моделирования значений этих единиц достаточно для покрытия бóльшей части всех словоупотреблений корпуса. Если распределение частот лексических единиц близко к нормальному, то, в соответствии с законом Парето, моделирования значений 20% самых частотных лексических единиц, т.е. имеющих большее число употреблений и соответственно высокую встречаемость в корпусе достаточно для покрытия 80% всех словоупотреблений корпуса. Если распределение в выборке отклоняется от нормального, соотношение может отличаться.

Выводы к главе 1.

Итак, в первой главе были проанализированы основные понятия: синтаксический анализ, сирконстанты, неоднозначные конструкции, стрелочная омонимия, стрелочная омонимия в конструкциях с сирконстантами, онтология, онтологическая семантика. В рамках данной работы синтаксический анализ понимается «процесс выявления иерархической структуры подчинительных связей предложения естественного языка, в частности русского, в соответствии с грамматикой языка. Результатом синтаксического анализа является дерево подчинительных связей (дерево синтаксического разбора)» [Окатьев, Гергель 2008]; синтаксически неоднозначными называются такие конструкции, в которых существуют неоднозначно устанавливаемые связи. Под сирконстантами понимаются, по определению [Теньер 1988:117], «обстоятельства (времени, места, способа и пр.), в которых развертывается процесс», они заполняют синтаксическую валентность глагола и являются факультативными членами предложения, что может привести к стрелочной омонимии. Стрелочная омонимия трактуется

как тип синтаксической неоднозначности, суть которой, по определению [Гладкий 1985: 113] состоит в том, что «некоторая НС (непосредственная составляющая) при переходе из одной структуры к другой «перевешивается», т.е. меняет хозяина». Далее были разобраны основные конструкции со стрелочной омонимией, где присутствуют сирконстанты и выбраны нужные типы конструкций, где «перевешивающая» НС представлена либо предложной, либо наречной группой или деепричастным оборотом. Был определен метод устранения неоднозначности средствами онтологической семантики, которая понимается как теория значения в естественном языке, которая использует конструирующую модель или же онтологию для извлечения значений и смыслов (см определение [Nirenburg 2004: 6]).Было дано определение онтологии, в частности, лингвистической, и описано ее устройство на примере онтологии AIIRE. В заключение главы было описано использование закона Парето в компьютерном моделировании значений лексических единиц в корпусе текстов. Этот закон будет применен в практической части для составления репрезентативной выборки полученных из корпуса неоднозначных конструкций.

Глава 2. МЕТОДЫ РАЗРЕШЕНИЯ СИНТАКСИЧЕСКОЙ НЕОДНОЗНАЧНОСТИ

В данной главе будет дан обзор на различные методы разрешения синтаксической неоднозначности, особое внимание будет уделено методу онтологической семантики; будут подробно разобраны основные принципы работы и структура универсального лингвистического процессора AIIRE, встроенной в него онтологии и ее инструментов; так же будет дано обоснование его выбора. Будет описан метод оценки эффективности выбранного метода.

2.1 Виды методов разрешения синтаксической неоднозначности

Существует несколько способов решения проблемы синтаксической неоднозначности.

Например, возможно решение данной задачи с использованием методов МО (машинного обучения). В работе [Yusuke Miyao, Jun'ichi Tsujii 2003] представлен метод разрешения синтаксической неоднозначности, основанные на лексикализованных грамматиках. Суть метода заключается в том, что модель выбирает самый вероятный результат синтаксического анализа из списка кандидатов, допустимых с точки зрения правил грамматики. «Поскольку результаты синтаксического анализа, представленные лексической грамматикой, не могут быть разложены на независимые подэтапы, мы применяем модель максимальной энтропии для лесов признаков, которая позволяет вероятностное моделирование без предположения о независимости. Наш подход обеспечивает общий метод создания последовательной вероятностной модели результатов анализа, данных лексикализированными грамматиками» [Yusuke Miyao, Jun'ichi Tsujii 2003]. Предложенная в работе модель включает в себя синтаксическую и семантическую вероятности, которые представляют синтаксические и семантические параметры соответственно. Поскольку последовательность синтаксических категорий ограничивает возможную структуру результатов анализа, вероятность семантики является условной вероятностью без разложения на примитивные зависимости слов. В работе используется метод максимальной энтропии для оценки вероятностей. Эффективность метода была доказана в ходе эксперимента.

В работе [Khalil Sima'an 2003] рассматриваются методы оптимизации вероятностных моделей для разрешения неоднозначности путем учета лингвистически важных метрик оценки. «Впоследствии мы представляем новые алгоритмы, которые оптимизируют метрики сопоставления деревьев, которые учитывают все более лингвистически значимые особенности

деревьев» [Khalil Sima'an 2003]. В данной работе, как и в предыдущей, рассматривается метод, сочетающий лингвистический подход и МО.

В статье [Jakub Zavrel at al. 1997] описывается метод Memory-Based Learning для разрешения синтаксической неоднозначности в конструкциях с предложными группами и проводится его сравнение с статистическими методами. Доказывается эффективность предложенного метода обучения. Memory-Based Learning относится к классу ленивых методов обучения, которые используют аналогии. В некоторых языках, по мнению авторов, задачу обработки «обычно можно описать только как сложное взаимодействие закономерностей, субрегулярностей и (семейств) исключений, хранение всех эмпирических данных как потенциально полезных при аналоговой экстраполяции работает лучше, чем извлечение основных закономерностей и отсутствие учета отдельных примеров» [Jakub Zavrel at al. 1997]. В работе используются метрики k-ближайших соседей и расчет подобия. В заключение доказывается, что подход MBL очень компетентен в разрешении неоднозначностей в конструкциях с предложными группами; «он обеспечивает лучшую производительность обобщения, чем многие предыдущие статистические подходы. Более того, поскольку мы можем измерить значимость функций, используя метрику увеличения информации IB1-IG, мы можем добавлять функции без больших затрат при выборе модели или комбинаторного взрыва. Дополнительным преимуществом подхода MBL является то, что, в отличие от других статистических подходов, он основан на использовании рассуждений по аналогии. Следовательно, это позволяет экспериментировать с различными типами распределенных несимвольных лексических представлений, извлеченных из корпусов с использованием обучения без контроля» [Jakub Zavrel at al. 1997].

Помимо методов МО, для решения задачи разрешения синтаксической неоднозначности могут так же использоваться лингвистические методы.

В [Добров 2016] дается обзор на существующие подходы к снятию неоднозначности с использованием онтологии или же без нее. Существуют синтаксические анализаторы, особенно одноцелевые парсеры, такие как DictaScope, STAGE-3, Stanford NLP, RASP, OpenNLP, которые выполняют синтаксический анализ, но не гарантируют правильность версий; Более того, поскольку они одноцелевые, они возвращают только наиболее вероятную версию. Основной вывод этой статьи состоит в том, что одним из наиболее эффективных способов достижения наиболее эффективного решения проблемы устранения неоднозначности является использование специальных сложных систем. Такие системы основаны на полном описании языка, и одним из них является онтология; поскольку можно эффективно решить проб лему синтаксической неоднозначности, используя другие уровни лингвистического анализа, в частности семантику и иногда прагматику.

Существуют различные примеры устранения синтаксической неоднозначности посредством других уровней лингвистического анализа. Например, в работе [Jurafsky 1996] основная идея заключалась в том, что применялся алгоритм, основанный на параллельном парсере. Парсер ранжировал конструкции и их интерпретации по условной вероятности таким образом, что менее вероятные конструкции не принимались во внимание. «Низкоранговые конструкции и интерпретации удалялись с помощью лучевого поиска»; или в работе [Johansson, Vogue 2008], где «система опирается на синтаксический и семантический подкомпоненты», и также было доказано, что «интеграция синтаксического и семантического анализа выгодна для них обоих».

Есть также несколько способов выполнить устранение синтаксической неоднозначности с помощью онтологии. Например, в работе [Гаранина 2016] представлена мультиагентная система, в которой агенты разрешают неоднозначности, вычисляя мощность их контекста и силу доказательств. Авторы утверждают, что данная информационная система может

использоваться для решения лексической неоднозначности и что она также является основой для подхода к снятию синтаксической неоднозначности, зависящему от контекста. В работе [Cimiano, Reyle 2003] есть еще один метод разрешения синтаксической неоднозначности. Авторы используют подход Muskens Logical Description Grammar (LDG), который позволяет унифицировать синтаксис и семантику единообразным образом в форме логических описаний»; они расширяют эту модель, «включая лексическую / онтологическую информацию и показывая, как это расширение может применяться к разрешению лексической неоднозначности».

В дополнение к работам, в которых разрешается синтаксическая неоднозначность, необходимо также упомянуть те, в которых лексическая неоднозначность разрешается с использованием синтаксического анализа и семантической интерпретации синтаксических структур. Так, в [Gelbach 1998] была предложена процедура «определения меры близости двух слов в словаре семантической сети». В некоторых случаях разрешение неоднозначности с использованием только синтаксиса невозможно, поэтому автор предлагает рассчитать пути между словами в семантическом словаре от одного слова до другого и выбрать наименее длинный путь в качестве решения.

Существует также еще один пример решения проблемы синтаксической неоднозначности с использованием уровня семантики. В [Ylonen 2011] «неоднозначности в выражении на естественном языке интерпретируются путем совместного устранения неоднозначности множественных альтернативных синтаксических и семантических интерпретаций».

Далее будет обоснован выбор ресурса для решения поставленной в работе практической задачи. Основываясь на том, что было сказано выше (см. 1.4), можно заключить, что одним из важных критериев выбора ресурса для автоматического разрешения стрелочной омонимии с сирконстантами

является наличие подробного описания валентностей, так как именно с их помощью и реализуется выбранный метод. Кроме того, необходимо наличие связей вышестоящих уровней с нижестоящими, для того чтобы сформулированные один раз ограничения на верхнем уровне потом применялись и на нижних. Для выполнения данной работы был выбран лингвистический процессор AIIRE и встроенная в него онтология потому, что этот ресурс соответствует вышеуказанным требованиям.

Как отмечает А.В. Добров, «Лингвистический процессор AIIRE представляет собой реализацию полномасштабного процесса NLU (Natural Language Understanding), основанного на методе межуровневого взаимодействия и методе снятия неоднозначности, основанного на правилах» [Добров 2014: 147].

Метод межуровневого взаимодействия, согласно А.В. Доброву, «заключается в том, чтобы избавиться от искусственного разделения уровней лингвистического анализа и произвести анализ морфологии, синтаксиса и семантики в одно и то же время. Этот способ анализа позволяет снимать неоднозначность на более низких уровнях с использованием правил верхнего уровня сразу же после возникновения неоднозначности на более низких уровнях, а не после анализа всего текста (или предложения) на этих уровнях» [там же]. Кроме того, данный принцип помогает минимизировать проблему комбинаторного взрыва, что очень важно для программного обеспечения NLP [Добров и др. 2017].

Корпус-менеджер AIIRE «включает в себя поддержку для корпусов на различных языках документов корпусов и включает ряд инструментов для автоматической разметки корпуса и обнаружения фрагментов этой разметки, требующих усовершенствований лингвистического обеспечения («ошибок» разметки)» [там же]. Таким образом, можно создать и обработать свой корпус из неоднозначных конструкций.

Отдельным инструментом работы с онтологией является Ontohelper, в интерфейсе которого моделируются значения и связи конкретно глаголов, для того, чтобы упростить процедуру прописывания их значений и связей. Остальные части речи моделируются в интерфейсе онтологии. Для них возможно создание вспомогательных интерфейсов в скором времени. Алгоритм работы в Ontohelper будет дан ниже в разделе 2.2.1.

Идея ontohelper строится на гипотезе, что любая (субъектная, объектная, дативная, предложная) валентность любого значения может быть:

1. выражена в виде одного и только одного базового класса онтологических концептов для значений существительных;

2. сама по себе задает один и только один базовый класс онтологических концептов для значений глаголов.

В общем виде данная гипотеза определяет любые валентности как отношения между базовыми классами онтологии. В этом заключается ключевое отличие онтологии AIIRE от ТКС, о которых говорилось выше (см. параграф 1.4.1).

В отличие от таких источников, как, например, семантический словарь Леонтьевой, онтологии не является «в основном зависимыми от языка». Онтология AIIRE содержит не только лексические значения, но также понятия, которые не могут быть привязаны к конкретным лексическим сущностям или даже к выражениям любых других языков» [Добров 2014].

Устранение неоднозначности в онтологии означает достижение правильного синтаксического анализа конкретной конструкции, где в результате будут получены правильные синтаксические деревья. Можно дополнительно уменьшить количество различных интерпретаций для конструкций, возникших в результате синтаксического анализа либо с использованием существующих семантических ограничений, либо путем

добавления новых (создание новых понятий, связей между ними), если этого требует конкретная конструкция. В конечном итоге, при условии, что все сделано правильно, каждая конструкция будет иметь правильные варианты анализа, среди которых не будет явно неправильных или абсурдных.

Так же в данном разделе необходимо обосновать, почему существующие тезаурусы для русского языка, например, такие как YARN, RussNet не могут быть эффективными в рамках решения поставленной в работе задачи.

В публикациях, посвященных проекту RussNet, упоминается информация о том, что в ходе разработки учитывались и прописывались валентности глаголов: «При построении тезаурусов мы учитывали закономерности морфолого-синтаксического оформления контекстов для разных значений знаменательных слов, в первую очередь глаголов. Здесь особое значение имеет характерная для русского языка реализация глагольных валентностей через предложные конструкции. В тезаурусе RussNet синсеты дополнены лексико-грамматическим описанием валентностей — статистически устойчивых параметров корпусных контекстов, в том числе сочетаний с предлогами» [Азарова и др. 2018]. Однако более подробной информации представлено не было. В более ранней работе [Азарова и др. 2004] приводились примеры рамок валентностей для глагола «направится» и его самых частотных значений, например, значения «двинуться в каком-либо направлении»: употребление в этом значении предполагает 2 обязательные валентности: (1) упоминание лица (группы лиц), которое совершает движение, причем, как правило, конкретный способ передвижения указан в непосредственной близости от данного (часто в составе того же самого предложения); (2) направления движения, которое представлено конструкцией «к + N3» (44%) (к дивану, к другу, к спуску, к нему…), называющей чаще (36%) место локализации, а реже (8%) – лицо (лиц), по направлению к которым ориентировано движение; в небольшом

числе случаев происходит сочетание этих частотных поверхностных структур (локализация + лицо) [там же]. Так же в выпускной квалификационной работе Годгильевой М.М. упоминалось про наличие в RussNet валентных рамок: «В RussNet у глагольного синсета есть список рамок валентностей с указанием на то, какая рамка соответствует какому члену синсета. Отмечается грамматическая форма аргументов, их порядок и факультативность, наличие или отсутствие предлога, семантические характеристики. Рамка валентностей наследуется от гиперонима к гипониму. Более того, валентность помогает различать значения многозначных глаголов» [Годгильева 2017]. Более конкретной информации или же ссылок в этой работе представлено не было.

На сайте проекта говориться о том, что «Частично синсеты снабжены контекстной информацией в виде рамок валентностей; формат их описания пока не полон. В качестве формата представления используется XML, однако значительная часть данных, подготовленных ранее, требует проверки и преобразования в актуальный формат». [Электронный ресурс] RussNet URL:http://ct05647.tmweb.ru/russnet/(дата обращения: 06.05.2020). Так же даются примеры синсетов в разделе «Синсеты RussNet» , один из которых можно скачать. Однако на сайте не представлено никакой информации по поводу того, как получить доступ к целому тезаурусу. На сайте [Электронный ресурс] WordNet URL:www.wordnet.ru (дата обращения: 06.05.2020), ссылка на который была отправлена рецензентом во время проверки статьи на основе данной работы для публикации в материалах конференции CompLing2020, были ссылки на скачивание баз данных Russian WordNet1.7.1. и Russian WordNet 3.0 для русского языка, вместе с которыми предлагалось скачать сам WordNet. Скачать последний не удалось по причине ошибки «Page not found» [Электронный ресурс] WordNet URL:https://wordnet.princeton.edu/obtain дата обращения 06.05.20).

Помимо RussNet существует YARN (Yet Another RussNet), информация о котором дается на том же сайте [Электронный ресурс]WordNet URL:http://ct05647.tmweb.ru/russnet/ (дата обращения: 06.05.20). На странице [Электронный ресурс] YARN URL:https://russianword.net/ дата обращения 06.05.20) указано, что «Работа над проектом прекращена с 2018 г.», а сам Yarn можно скачать либо в формате xml файла, либо в формате csv файла. При попытке скачать файл формата xml возникает ошибка, и страница падает. [Электронный ресурс] YARN URL:https://russianword.net/yarn.xml (дата обращения: 06.05.20) Файл в формате csv содержит синсеты, иногда с частеречными тегами, но никакой информации о валентностях там нет.

Стоит также отметить, что английские источники, такие как SUMO, OpenCyc, WordNet, не были приняты во внимание в этой работе, поскольку вышеупомянутые проекты были сделаны специально для английского языка, в то время как рассматриваемая проблема касается русского языка.

2.2 Разрешение синтаксической неоднозначности средствами онтологической семантики

В данном параграфе будет описан метод разрешения неоднозначности средствами онтологической семантики.

Чтобы разрешить стрелочную омонимию, необходимо правильно трактовать связь внутри неоднозначной конструкции, определить, кому из двух потенциальных хозяев принадлежит сирконстант.

Как уже упоминалось в разделе 1.4, для решения поставленной задачи надо учесть валентности слов, которые реализуются в онтологии. Метод онтологической семантики заключается в том, чтобы обеспечить семантические связи между словами, путем моделирования соответствующих отношений.

Моделирование в онтологии слов происходит в соответствие с правилами конвенций и методики по работе с онтологией (см раздел 1.4.2): слова выделяются из неоднозначных конструкций и далее вручную моделируются, то есть прописываются, их значения. Например, для слова «человек» есть значение «конкретная особь вида homo sapiens». Оно представляет собой класс «homo sapiens», который обладает обобщенными атрибутами, свойственными всем его представителям, например, умение общаться посредством языка. От своего гиперонима «человек (конкретная особь вида homo)» наследует следующий атрибут: отношение «изготовлять (производить продукт)» с объектом отношения «орудие труда (предмет, используемый человеком для осуществления какой-либо деятельности)». При этом у этого рассматриваемого понятия так же есть атрибут «жить (осуществлять жизнедеятельность)», который он наследует уже от понятия «организм (живое существо)». Через гиперонимы «человек» наследует связи с базовыми понятиями, такими как, например «живое существо». С помощью атрибутов формулируются возможные валентности для данного слова. Так, «человек» является субъектом отношения «участие в роли субъекта действия», то есть может быть носителем какого-либо действия и делать что-либо. Это прямое бинарное отношение, которому соответствует обратное отношение «обладать субъектом действия», то есть какое-либо действие может совершаться «человеком». Таким образом, благодаря тому, что такое отношение прописано в онтологии, оно будет распознано при автоматическом синтаксическом разборе предложения или словосочетания, где идет речь о человеке, совершающем какое-либо действие.

Особое место среди классов занимают базовые классы — это верхние классы в иерархии понятий, а также важнейшие классы вроде класса «человек». Базовые классы, как правило, обладают значительным числом отношений разных типов. (Конвенции и методики по работе с онтологией [Электронный ресурс] Онтология AIIRE URL:http://ontology.aiire.org/static/conventions.html (дата обращения: 05.09.2019)). Экземплярами называют

конкретные объекты, которые принадлежат классу [там же]. Например, «А.С. Пушкин» является экземпляром класса «российский поэт XIX века». Экземпляры отличаются от классов тем, что не вступают между собой в родовидовые отношения [там же].

При наследовании может происходить замещение атрибутов, которое заключается в том, что наследуемые отношения и/или объекты этих отношений замещаются на свои гипонимы [там же]. Например, понятие «отрезок времени» обладает отношением «длительность», а понятие «отрезок» — отношение «длина отрезка». Первое понятие является гипонимом последнего, следовательно, атрибут «длительность» будет гипонимом атрибута «длина отрезка».

После моделирования значений можно загрузить конструкции целиком. «КМ позволяет загружать неразмеченные тексты или тексты вертикальном формате для их дальнейшей автоматической разметки, отражающей структуры непосредственных составляющих и зависимостей, при этом единицы, ранее считавшиеся токенами, разбиваются на алломорфы, которые далее объединяются в древовидные структуры» [П. Л. Гроховский и др. 2017]. Они будут автоматически обработаны лингвистическим процессором на основе прописанных в онтологии правил.

Например, рассмотрим неоднозначную конструкцию «выдвижение кандидатов в депутаты по избирательным округам», где возможны следующие интерпретации:

•«выдвижение кандидатов в депутаты по направлению к избирательным округам»

•«кандидаты в депутаты двигаются по избирательным округам»

•«кандидаты в депутаты по избирательным округам куда-то ездят»

Кроме того, возможны различные комбинации этих вариантов, которые увеличат общее количество синтаксических деревьев. Если синтаксический анализ выполнен правильно, то останется только один правильный вариант синтаксического анализа. Корректный анализ в этом случае означает, что «кандидаты» распознаются лингвистическим процессором как целостное понятие, которое регулируется существительным «выдвижение», и что «выдвижение» может происходить «от группы избирателей».

Синтаксические связи между словами прописываются в деревьях, а соответствующие семантические отношения между концептами – в семантических графах. Чтобы понять, какие версии верны, а какие нет, необходимо сравнить их друг с другом и устранить ошибочные путем вторичного моделирования понятий в Ontohelper и онтологии.

2.2.1 Инструменты и алгоритмы разработки онтологии

Для моделирования понятий в онтологии используется следующий алгоритм:

1. Проверить наличие понятия в онтологии, если оно отсутствует — добавить его.
2. Уточнить при помощи контекстов из полученной на основе синтаксического подкорпуса НКРЯ выборки неоднозначных конструкций и Викисловаря [Электронный ресурс] Викисловарь//URL:https://ru.wiktionary.org/wiki/%D0%97%D0%B0%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D0%BD%D0%B8%D1%86%D0%B0 (дата обращения: 05.09.2019), в каких значениях употребляется лексическая единица, соответсвующая понятию, проверить наличие этих значений в онтологии; если нужное значение отсутствует — добавить его.

3. Моделировать значение в соответствии с требованиями конвенций и методики работы с онтологией [Электронный ресурс] Конвенции и методики по работе с онтологией URL:http://ontology.aiire.org/static/conventions.html (дата обращения 05.09.2019).

Согласно конвенции для каждой части речи существуют свои алгоритмы моделирования в онтологии. Существительные обрабатываются следующим образом:

1. Указать вышестоящие понятия (гиперонимы), обеспечить связь с базовыми понятиями; в случае, если гиперонимы уже добавлены, проверить, являются ли они корректными, и убедиться, что они обработаны в соответствии с вышеуказанными конвенциями и имеют привязку к вышестоящим классам; после таким же образом проверить нижестоящие понятия (гипонимы);
2. Проверить или указать синонимы; под синонимами в конвенциях понимаются слова, которые обозначают то же понятие, что и обрабатываемое существительное; убедиться, что все синонимы корректны, и обработаны в соответствие с вышеуказанной конвенцией;
3. если для существительного существует соответствующее действие, выражаемое глаголом, то перейти к моделированию глагола и добавить к нему существительное в качестве процесса, который соответствует действию;
4. если у существительного есть или может быть добавлен соответствующий ему атрибут — добавить атрибут или убедиться, что он корректен и обработан по правилам конвенции;
5. если необходимо указать генитивное отношение — оформить данное отношение по методике для отношений.

Необходимость указания атрибутов или генитивных отношений определяется их наличием у рассматриваемого существительного в выборке.

Глаголы обрабатываются в ontohelper по следующему алгоритму:

1. зайти в ontohelper, выбрать глагол и его значение, если глагола или нужного значения нет — добавить;
2. определить тип глагола: «действие», «состояние» или «процесс»
3. если был выбран тип «действие», указать соответствующую форму совершенного вида с нужным значением;
4. иначе указать только процессуальное существительное, соответствующее глаголу;
5. если процессуальное существительное или форма совершенного вида у глагола отсутствуют, добавить техническую форму или техническое процессуальное существительное;
6. указать субъект действия;
7. указать направленность действия;
8. указать адресованность действия.

Прилагательные обрабатываются по отдельной методике, указанной в [Электронный ресурс]// Онтология: методика URL:https://docs.google.com/document/d/1 77ZqkCT0MfDV8zq6HkpSYWwM0S_8lpsNYta3gk5IbM/edit?skip_itp2_check=true# (дата обращения: 05.09.2019):

1. установить, какое свойство обозначает прилагательное;
2. свойство должно быть выражено существительным и обработано по методике для существительных;
3. установить, какой сущности (какому объекту или процессу) принадлежит это свойство (выявляется тем же способом), тоже обработать в онтологии;
4. создать прямое и обратное атрибутивные отношения между сущностью и свойством, при этом важно, чтобы отношения устанавливались между абстрактными, а не конкретными свойством и сущностью;

5. выбрать «атрибутивное отношение, а в качестве объекта указать гипоним свойства, соответствующий значению прилагательного» [там же];
6. «Если у указанного гипонима свойства есть какие-либо гиперонимы по дороге к свойству, то создать гиперонимы прилагательного в соответствии с указаниями интерфейса» [там же];
7. «В противном случае указать для прилагательного в качестве гиперонима «атрибут X» [там же] или «переменный атрибут X», если есть существительное на – ость.

Для предлогов в методике указан следующий алгоритм [там же]:

1. все предлоги обозначают предложные отношения;
2. большинство предложных отношений являются отношениями между значением глагола и значением существительного (приглагольные предлоги, ср. *сидеть в мешке*), меньшая часть значений предлогов — это отношения между значениями двух существительных (прииенные предлоги, ср. *кот в мешке*);
3. для приглагольного предлога указать название класса субъекта отношения в предложном падеже, обозначить характер связи между субъектом и объектом отношения в форме деепричастной группы или, в случае невозможности такой формулировки, в формулировке «по отношению к», указать название класса объекта отношения в падеже, требуемом характером связи;
4. Предложные отношения классифицируются так же, как и все остальные отношения;
5. Для предложных отношений необходимо явное указание субъектов и объектов через технические отношения «обладание субъектом отношения» и «обладание аргументом предложного отношения», соответственно. Эти отношения, как и любые другие, могут быть указаны либо в самом предлоге, либо в любом его надклассе.

Наречия обрабатываются по аналогии с прилагательными и могут быть связаны с ними через отношение «обладание атрибутом атрибута» [там же].

Для местоимений явно прописанного алгоритма в конвенциях или же в методике нет. На практике они обрабатывались по аналогии с существительными или же с прилагательными, но менее детально: указывались значения и гиперонимы.

2.3 Метод построения выборки лексических единиц в соответствии с законом Парето

Для построения выборки лексических единиц необходимо сделать следующее.

1. Построить частотные словари: для каждой словоформы в составе неоднозначной конструкции определить ее часть речи и записать ее лемму в соответствующий словарь в качестве ключа. Значением по ключу будет количество употреблений словоформы в выдаче;
2. В каждом словаре посчитать общее количество словоупотреблений, которое равно сумме всех частот;
3. Выполнить сортировку ключей словаря (лексических единиц) по убыванию их частот;
4. Выполнить перебор отсортированных по убыванию частотности ключей словаря (лексических единиц), суммируя частоты из словаря до тех пор, пока сумма не будет равна нужному проценту от общего количества словоупотреблений, например, 80% в соответствии с законом Парето.

2.4 Метод оценки результатов

После получения результатов автоматического снятия неоднозначности их необходимо оценить и проанализировать следующим образом:

1. посчитать количество корректно разобранных конструкций, их соотношение к общему количество разобранных конструкций;
2. оценить соотношение ошибочных вариантов к корректным;
3. посчитать количество разобранных неверно конструкций и проанализировать причины ошибок;
4. дать общую оценку исследуемому методу, его эффективности и трудоемкости.

Выводы к главе 2.

Во второй главе были рассмотрены виды методов разрешения синтаксической неоднозначности на основе машинного обучения с использованием статистики, комбинированного подхода, который учитывает и МО, и лингвистику, чисто лингвистические подходы без использования онтологий и так же метод, предполагающий разрешение неоднозначности средствами онтологической семантики. Был подробно описан лингвистический процессор AIIRE, который, как было сказано выше (см. пар. 2.1) представляет собой реализацию полномасштабного процесса NLU, основанного на методе межуровневого взаимодействия и методе устранения неоднозначности на основе правил; был обоснован выбор именно этого процессора. Встроенная в него онтология позволяет прописывать все возможные языковые отношения, валентности глаголов, необходимые для решения поставленной в работе задачи снятия неоднозначности; онтология содержит не только лексические значения, как, например, семантические словари, но, так же, и понятия, которые позволяют онтологии не быть привязанной к конкретным лексическим сущностям. Далее были описаны

инструменты компьютерной лингвистической онтологии, в частности ontohelper, в котором прописываются валентности глаголов и общий принцип ее работы, который заключается в том, что в ходе анализа конструкции устанавливаются все возможные отношения между понятиями в ней и строятся соответствующие семантические графы и синтаксические деревья; Так же был описан метод оценки эффективности полученных результатов.

Глава 3. моделирование НЕОДНОЗНАЧНЫХ КОНСТРУКЦИЙ С СИРКОНСТАНТАМИ

В данной главе будут описаны практические действия, предпринятые для выполнения ранее поставленной задачи: составление поисковых запросов для поиска неоднозначных конструкций в синтаксическом подкорпусе НКРЯ, описание алгоритмов на языке Python для автоматической выгрузки конструкций, получения статистических данных о них, построения выборки лексических единиц, выбора конструкций на их основе и загрузки конструкций в корпус-менеджер процессора AIIRE. Далее будут показаны на конкретных примерах процедуры компьютерного моделирования значений и семантических валентностей. В конце главы будет выполнена оценка эффективности разрешения неоднозначности.

3.1Сбор данных из синтаксического подкорпуса НКРЯ

Для осуществления поставленной практической задачи было необходимо собрать корпус неоднозначных конструкций по составленным ранее запросам. Корпус собирался на основе синтаксического подкорпуса Национального Корпуса Русского Языка (НКРЯ), который был выбран потому, что в этом подкорпусе содержатся тексты с синтаксической и морфологической разметкой, что упрощает задачи выделения нужных

элементов конструкций на основе частеречных тэгов, получения лемм для каждой словоформы конструкции. Было необходимо получить по запросам максимальное количество неоднозначных конструкций, чтобы охватить как можно большее количество явлений и, тем самым, обеспечить максимально возможную достоверность полученных результатов. Ставилась задача составить частотные словари для каждой части речи каждого запроса в соответствии с методом, описанным во второй главе в разделе 2.4. Решение должно было включать в себя разработку алгоритма, позволяющего выгружать неоднозначные конструкции автоматически. Данное условие обосновано тем, что объем синтаксического подкорпуса НКРЯ составляет 650 документов, 66 684 предложения, 1 031 675 слов и, соответственно, сбор необходимого количества данных вручную не оптимален. Выгрузить данные с помощью сайта не удалось по техническим причинам: при попытках скачать данные в одном из предлагаемых форматов (Excel, OpenOffice Calc, XML) возникала ошибка 502 Bad Gateway.

3.1.1 Составление запросов в соответствии с типами конструкций

В первой главе были рассмотрены восемь типов неоднозначных синтаксических структур со стрелочной омонимией. На их основе были выделены более конкретные подтипы — конструкции со стрелочной омонимией с сирконстантами. Для каждого подтипа неоднозначной конструкции формулировались поисковые запросы.

Неоднозначность определяется тем, что сирконстант может быть зависим от одного из двух хозяев. Следовательно, вариантов запросов для каждого типа конструкции должно быть два: в первом сирконстант зависит от первого хозяина, во втором — от второго. В некоторых запросах также учитывались расстояния между элементами конструкции, чтобы передать линейный порядок элементов внутри конструкции. Для каждого подтипа представлена схема, которая отображает частеречный состав конструкции и линейный порядок элементов, который задается знаком «+»; также в схеме

жирным шрифтом в подтипах отображается хозяин, в самом типе жирным шрифтом выделен сирконстант.

В рамках данного исследования были выделены следующие типы.

1. Сирконстант здесь — это предложная группа **пр+сущ**, у которой может быть один из двух хозяев: гл и сущ – это конструкция типа 1 по А.В. Гладкому (см. параграф 1.3.1). Но также возможны варианты конкурирующих хозяев:

Подтип 1.1**.**: схемы

1. **гл+**сущ+пр+сущ (рис. 1)
2. гл+**сущ+**пр+сущ (рис. 2)

Здесь конкурирующие хозяева — сущ и гл. К данным схемам относятся разные структуры. Наиболее типичной является глагол + дополнение + предложная группа, однако может быть и глагол + подлежащее + предложная группа, как показано в примере 1.

Пример:

(1) «оказался вопрос о социализме» [Материалы Уппсальского корпуса, коллекция 716 2019]

Формы поиска для запроса 1.1 представлены на рисунках 1 и 2:

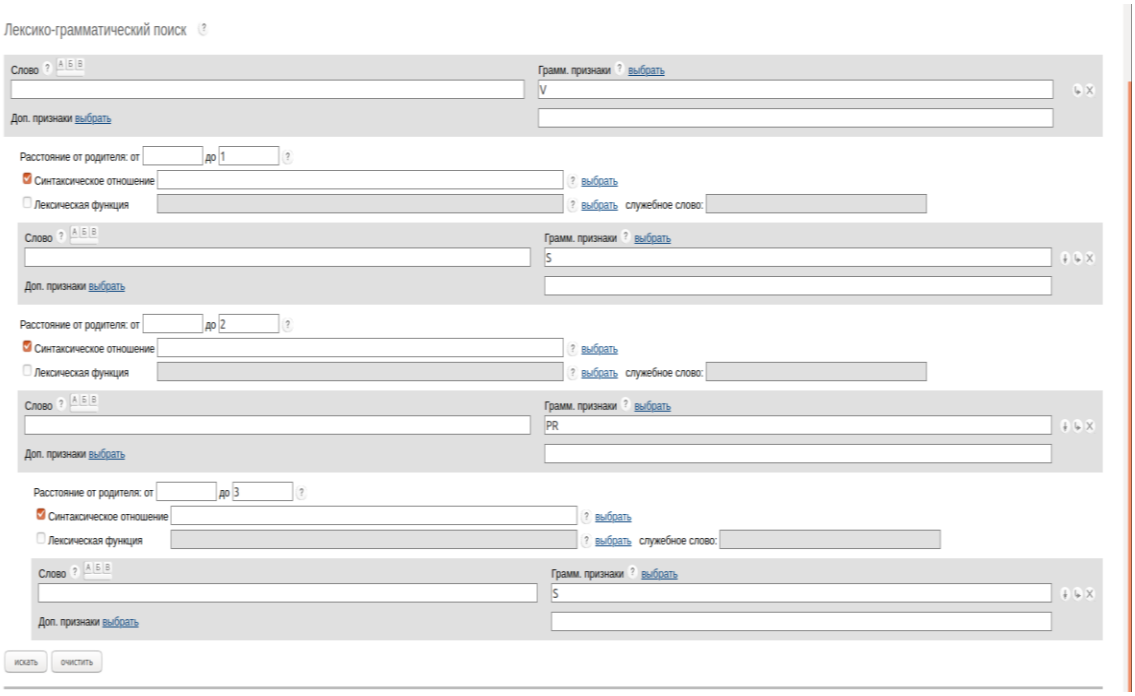


Рисунок 1. Форма запроса для схемы 1.

Сирконстант — предложная группа, которая зависит от глагола; линейный порядок элементов предложной группы задан с помощью расстояния — существительное находится на расстоянии 1 от хозяина, т.е. от предлога. Также от глагола зависит существительное. Линейный порядок элементов задан с помощью расстояний: глагол — хозяин, на расстоянии до 2 от глагола находится предложная группа, зависимое существительное уже может находиться между глаголом и предложной группой на расстоянии до 2 от глагола.

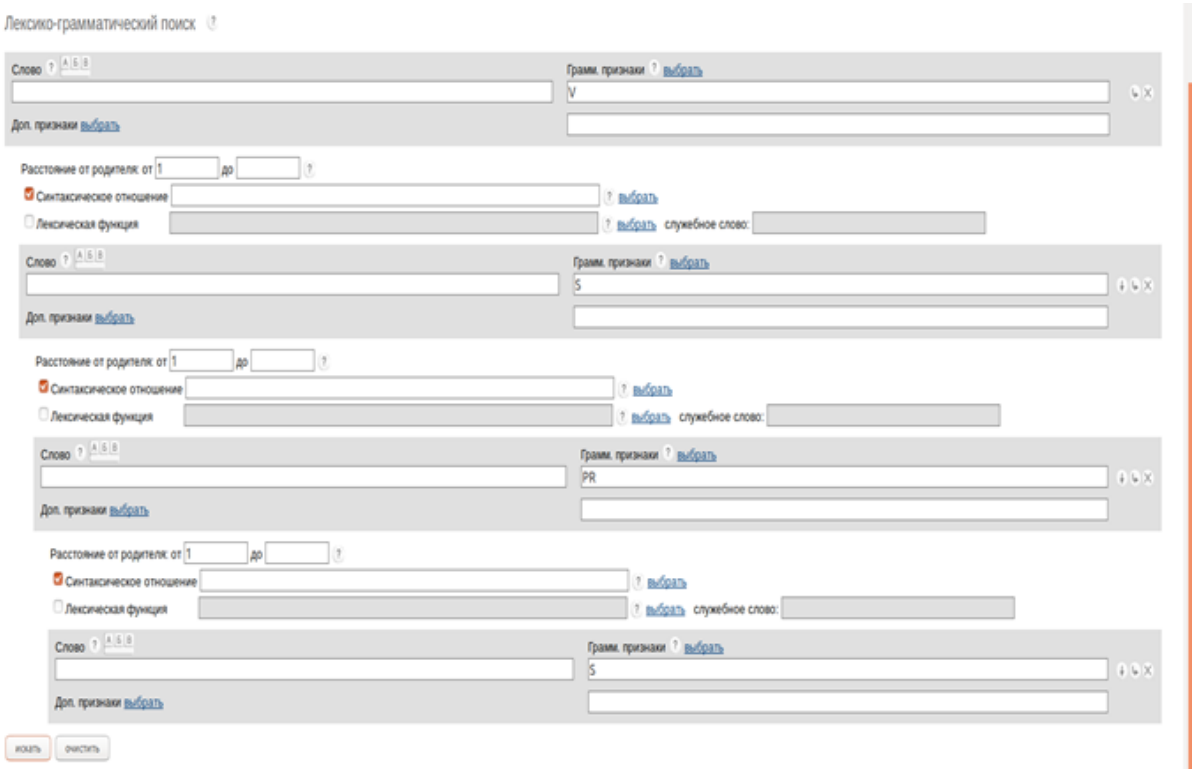


Рисунок 2. Форма запроса для схемы 2.

Сирконстант — предложная группа, которая зависит от существительного; линейный порядок элементов предложной группы задан с помощью расстояния — существительное находится на расстоянии 1 от хозяина, т.е. от предлога. Предложная группа зависит от существительного, а оно зависит от глагола. Линейный порядок элементов задан с помощью расстояний: глагол — хозяин, на расстоянии 1 от глагола находится существительное, на расстоянии от 1 от зависимого существительного находится предложная группа.

Представлены ссылки на запросы для подтипа 1.1:

Ссылка для запроса по схеме 1:

http://processing.ruscorpora.ru/search.xml?out=normal&kwsz=5&dpp=100&spd=10&spp=100&seed=24747&text=lexgramm&mysent=&level1=0&level2=1&level3=1&level4=2&type4=&flags2=&type3=&type2=&flags4=&flags1=&flags3=&mysize=&mysentsize=&simple=1&env=alpha&parent2=1&link4=on&link3=on&link2=on&gramm4=S&gramm1=V&gramm2=S&gramm3=PR&min2=&min3=&min4=&lang=ru&lex4=&lex1=&lex3=&max2=1&max3=2&lex2=&mycorp=&max4=3&notag=1&parent4=3&parent3=1&mode=syntax&parent1=0

Ссылка для запроса по схеме 2:

http://processing.ruscorpora.ru/search.xml?sort=i_grtagging&out=normal&dpp=100&spd=100&seed=3285&simple=1&text=lexgramm&mysent=&level1=0&level2=1&level3=2&level4=3&type4=&type3=&type2=&flags4=&flags1=&flags3=&mysize=&mysentsize=&parent4=3&env=alpha&mode=syntax&link4=on&link3=on&link2=on&gramm4=S&gramm1=V&gramm2=S&gramm3=PR&lex2=&min2=1&min3=1&min4=1&lang=ru&lex4=&lex1=&lex3=&max2=&max3=&flags2=&mycorp=&max4=&notag=1&parent3=2&parent2=1&parent1=0

Подтип 1.2 : схемы

1. **деепр**+сущ+пр+сущ(рис. 3)
2. деепр+**сущ**+пр+сущ(рис. 4)

это частный случай 1.1, здесь конкурирующие

хозяева: гл в форме деепр и сущ.

Пример:

(2)«разложив лапы на месте» [Н. Добрецов 2019]

Формы поиска для запроса 1.2 представлены на рисунках 3 и 4:

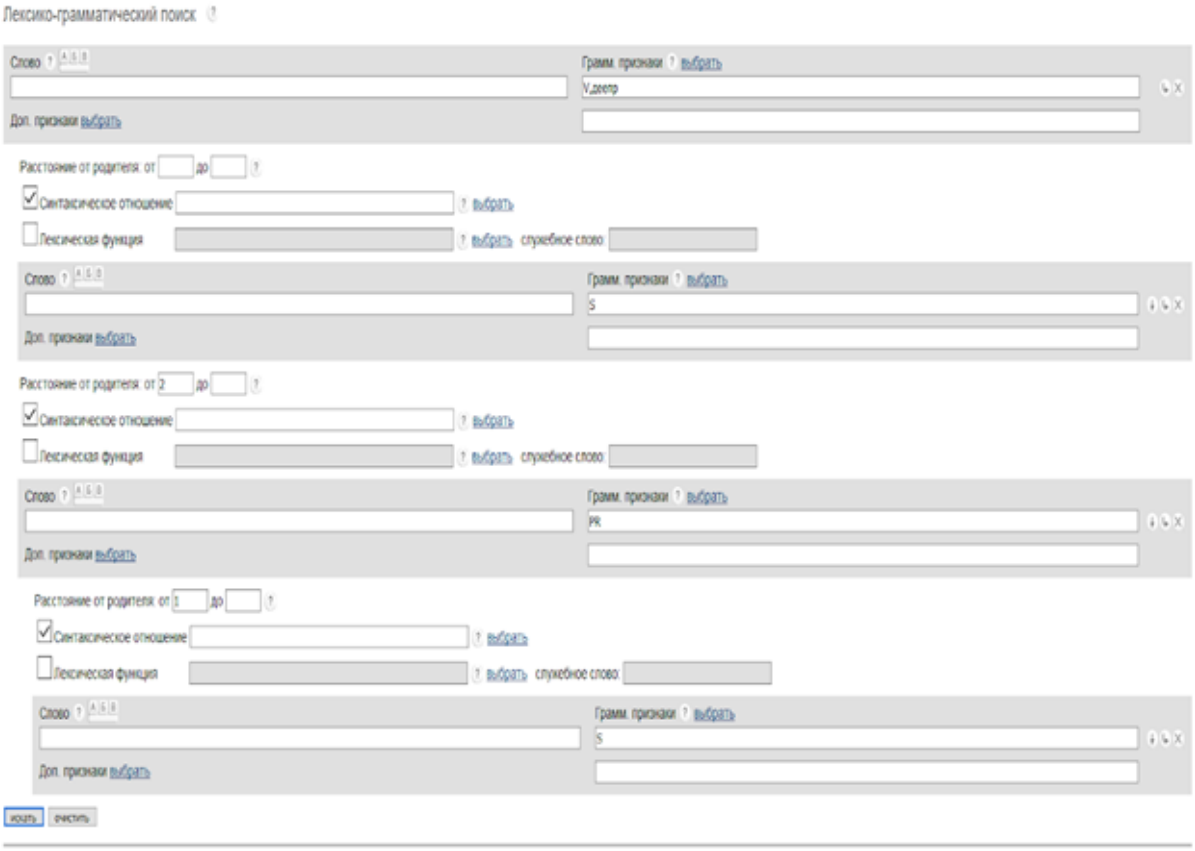


Рисунок 3. Форма запроса для схемы 1.

Сирконстант — предложная группа, которая зависит от глагола в форме деепричастия; линейный порядок элементов предложной группы задан с помощью расстояния — существительное находится на расстоянии 1 от хозяина, т.е. от предлога. Также от глагола в форме деепричастия зависит существительное. Линейный порядок элементов задан с помощью расстояний: глагол — хозяин, на расстоянии от 1 от глагола находится предложная группа, зависимое существительное уже может находиться между глаголом и предложной группой.

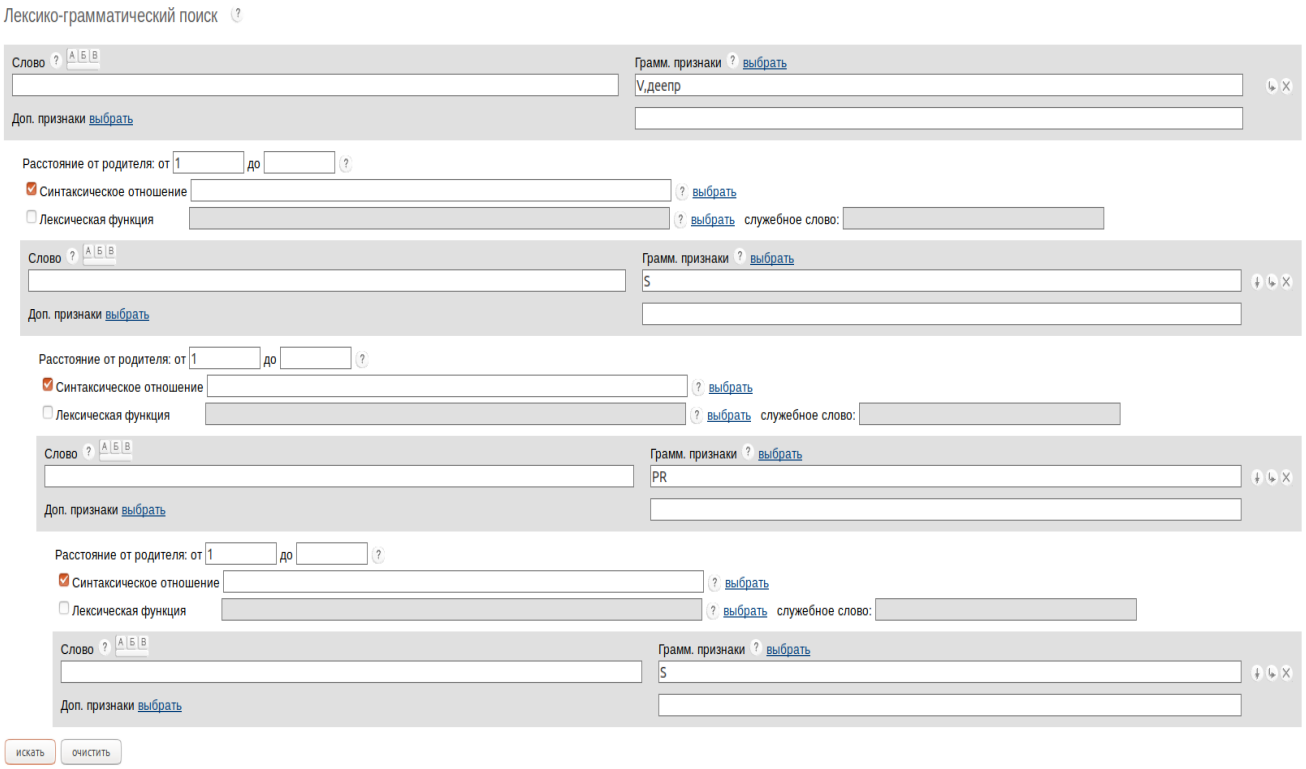


Рисунок 4. Форма запроса для схемы 2.

Сирконстант — предложная группа, которая зависит от существительного; линейный порядок элементов предложной группы задан с помощью расстояния — существительное находится на расстоянии 2 от хозяина, т.е. от предлога. Предложная группа зависит от существительного, а оно зависит от глагола в форме деепричастия. Линейный порядок элементов задан с помощью расстояний: глагол в форме деепричастия — хозяин, на расстоянии 1 от глагола находится сначала существительное, на расстоянии от 1 от зависимого существительного находится предложная группа.

Представлены ссылки на запросы для подтипа 1.2:

Ссылка для запроса по схеме 1:

http://processing.ruscorpora.ru/search.xml?sort=i_grtagging&out=normal&dpp=100&spd=100&seed=3549&text=lexgramm&mysent=&level1=0&level2=1&level3=1&level4=2&type4=&flags2=&type3=&type2=&flags4=&flags1=&flags3=&mysize=&mysentsize=&simple=1&env=alpha&parent2=1&link4=on&link3=on&link2=on&gramm4=S&gramm1=V%2C%D0%B4%D0%B5%D0%B5%D0%BF%D1%

80&gramm2=S&gramm3=PR&min2=&min3=2&min4=1&lang=ru&lex4=&lex1=&lex3=&max2=&max3=&lex2=&mycorp=&max4=&notag=1&parent4=3&parent3=1&mode=syntax&parent1=0

Ссылка для запроса по схеме 2:

http://processing.ruscorpora.ru/search.xml?sort=i_grtagging&out=normal&dpp=100&spd=100&seed=25646&text=lexgramm&mysent=&level1=0&level2=1&level3=2&level4=3&type4=&flags2=&type3=&type2=&flags4=&flags1=&flags3=&mysize=&mysentsize=&simple=1&env=alpha&parent2=1&link4=on&link3=on&link2=on&gramm4=S&gramm1=V%2C%D0%B4%D0%B5%D0%B5%D0%BF%D1%80&gramm2=S&gramm3=PR&min2=1&min3=1&min4=1&lang=ru&lex4=&lex1=&lex3=&max2=&max3=&lex2=&mycorp=&max4=&notag=1&parent4=3&parent3=2&mode=syntax&parent1=0

Подтип 1.3 : схемы

1. **сущ**+сущ+пр+сущ (рис. 5)
2. сущ+**сущ**+пр+сущ (рис. 6)

здесь конкурирующие хозяева сущ и сущ.

Пример:

(3)«программа переселения с земель»[В. Дубнов 2019]

Формы поиска для запроса 1.3 представлены на рисунках 5 и 6:

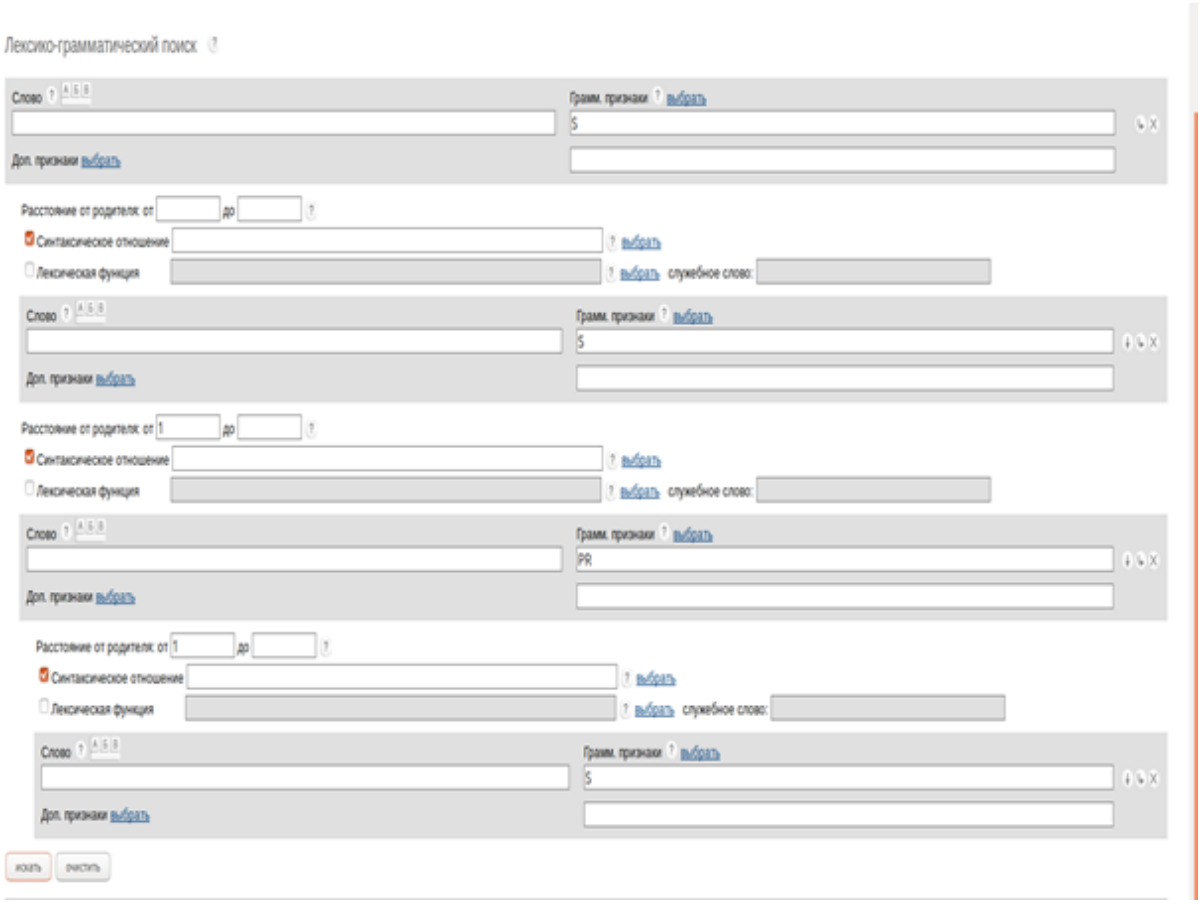


Рисунок 5. Форма запроса для схемы 1.

Сирконстант — предложная группа, которая зависит от существительного; линейный порядок элементов предложной группы задан с помощью расстояния — существительное находится на расстоянии 1 от хозяина, т.е. от предлога. От существительного — хозяина зависит еще одно существительное. Линейный порядок элементов задан с помощью расстояний: существительное — хозяин, на расстоянии от 1 от глагола находится предложная группа, зависимое существительное уже может находиться между глаголом и предложной группой.

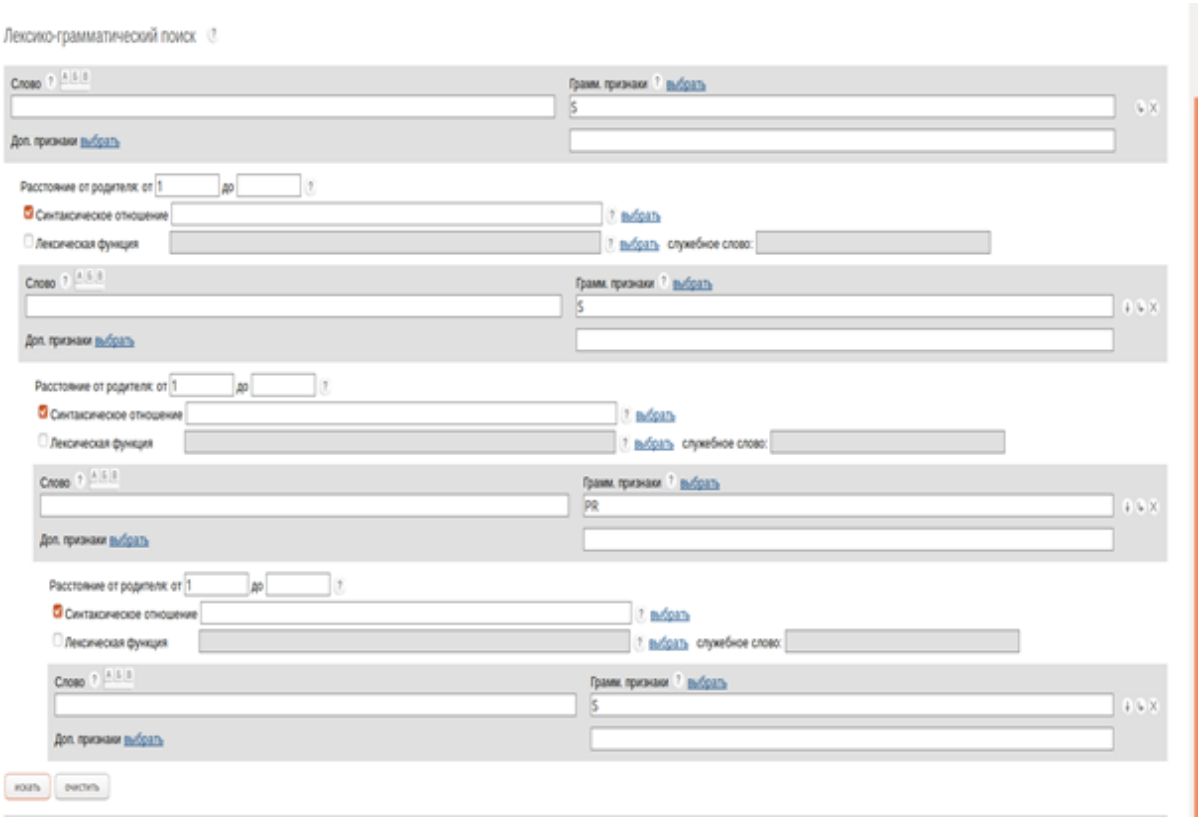


Рисунок 6. Форма запроса для схемы 2.

Сирконстант — предложная группа, которая зависит от существительного; линейный порядок элементов предложной группы задан с помощью расстояния — существительное находится на расстоянии 1 от хозяина, т.е. от предлога. Предложная группа зависит от существительного, а оно зависит от другого существительного. Линейный порядок элементов задан с помощью расстояний: существительное — хозяин, на расстоянии 1 от существительного — хозяина находится сначала существительное, на расстоянии от 1 от зависимого существительного находится предложная группа.

Представлены ссылки на запросы для подтипа 1.3:
Ссылка для запроса по схеме 1:

http://processing.ruscorpora.ru/search.xml?sort=i_grtagging&out=normal&dpp=100&spd=100&seed=29547&text=lexgramm&mysent=&level1=0&level2=1&level3=2&level4=3&type4=&flags2=&type3=&type2=&flags4=&flags1=&flags3=&mysize=&mysentsize=&simple=1&env=alpha&parent2=1&link4=on&link3=on&link2=on&gramm4=S&gramm1=S&gramm2=S&gramm3=PR&min2=1&min3=1&mi

n4=1&lang=ru&lex4=&lex1=&lex3=&max2=&max3=&lex2=&mycorp=&max4=&notag=1&parent4=3&parent3=2&mode=syntax&parent1=0

Ссылка для запроса по схеме 2:

http://processing.ruscorpora.ru/search.xml?sort=i_grtagging&out=normal&dpp=100&spd=100&seed=7127&text=lexgramm&mysent=&level1=0&level2=1&level3=1&level4=2&type4=&flags2=&type3=&type2=&flags4=&flags1=&flags3=&mysize=&mysentsize=&simple=1&env=alpha&parent2=1&link4=on&link3=on&link2=on&gramm4=S&gramm1=S&gramm2=S&gramm3=PR&min2=&min3=1&min4=1&lang=ru&lex4=&lex1=&lex3=&max2=&max3=&lex2=&mycorp=&max4=&notag=1&parent4=3&parent3=1&mode=syntax&parent1=0

Подтип 1.4 : схемы
сущ+прич+пр+сущ(рис. 7)
сущ+**прич**+пр+сущ (рис. 8)
здесь конкурирующие хозяева сущ и
прич.
Пример:
(4) «человек, подрастерявший в борьбе» [В. Дубнов 2019]

Формы поиска для запроса 1.4 представлены на рисунках 7 и 8:

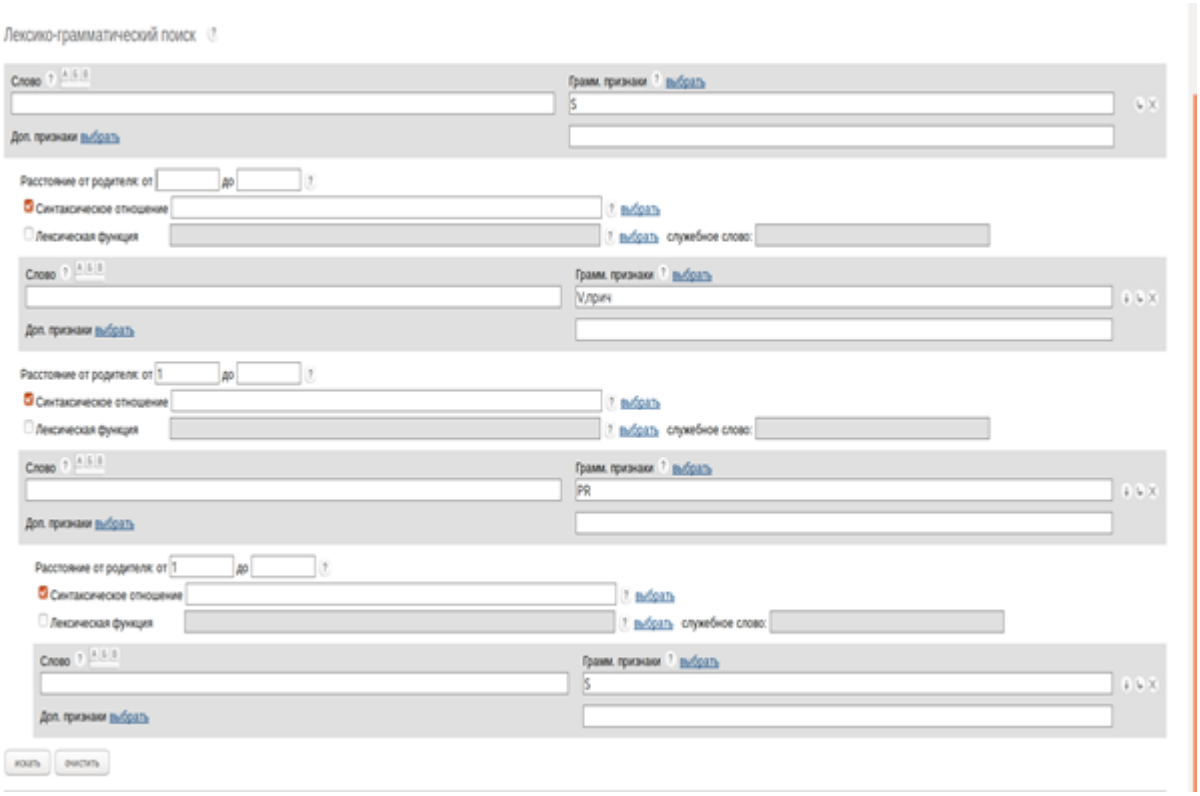


Рисунок 7. Форма запроса для схемы 1.

Сирконстант — предложная группа, которая зависит от существительного; линейный порядок элементов предложной группы задан с помощью расстояния — существительное находится на расстоянии 1 от хозяина, т.е. от предлога. Предложная группа зависит от существительного — хозяина, также от него зависит глагол в форме причастия. Линейный порядок элементов задан с помощью расстояний: существительное — хозяин, на расстоянии 1 от него находится предложная группа, глагол в форме деепричастия может находиться между существительным — хозяином и предложной группой.

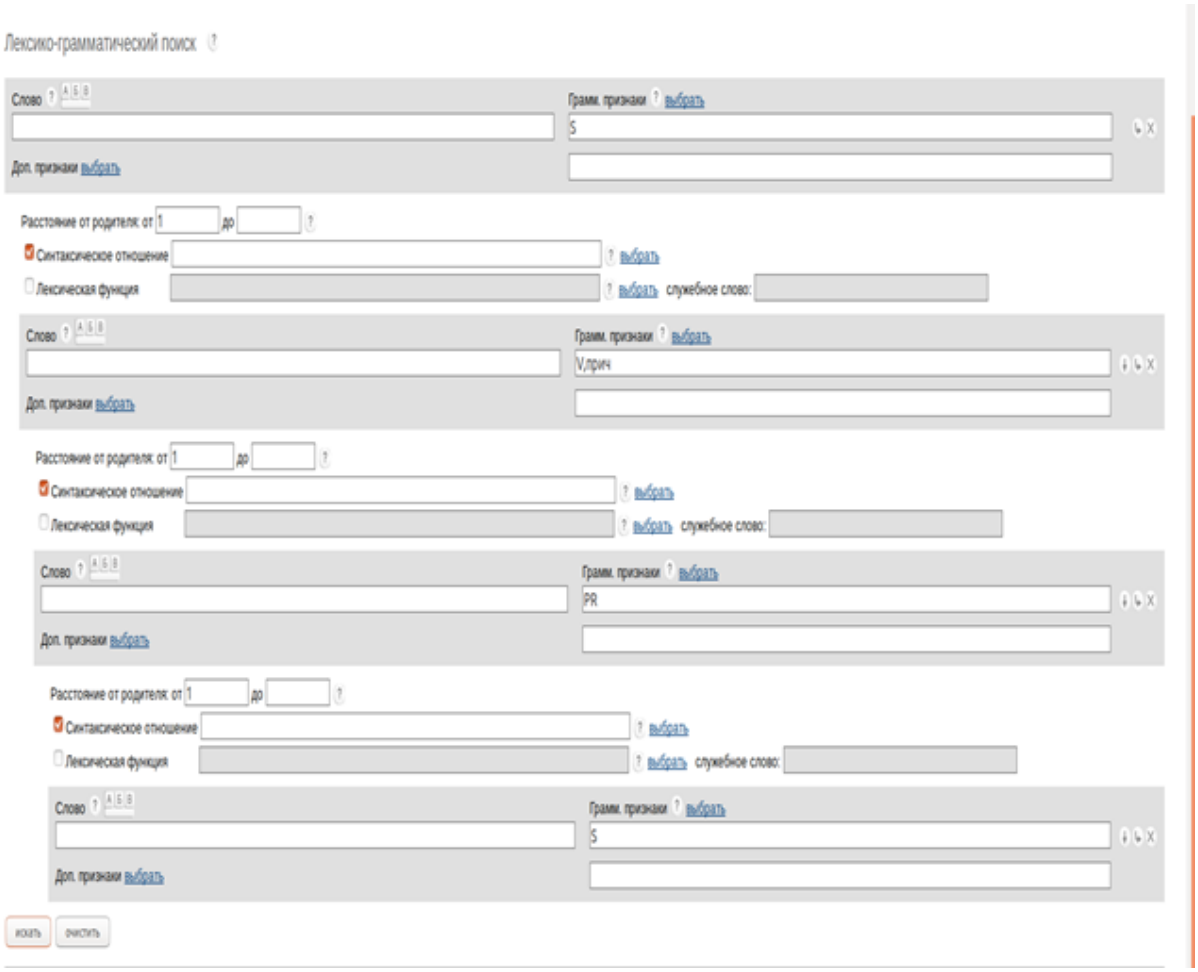


Рисунок 8. Форма запроса для схемы 2.

Сирконстант — предложная группа, которая зависит от глагола в форме причастия, который зависит от существительного — хозяина; линейный порядок элементов предложной группы задан с помощью расстояния — существительное находится на расстоянии 1 от хозяина, т.е. от

предлога. Линейный порядок элементов конструкции задан с помощью расстояний: существительное — хозяин, на расстоянии 1 от существительного — хозяина находится сначала глагол в форме причастия, на расстоянии от 1 от глагола в форме причастия находится предложная группа.

Представлены ссылки на запросы для подтипа 1.4:
Ссылка для запроса по схеме 1:
http://processing.ruscorpora.ru/search.xml?sort=i_grtagging&out=normal&dpp=100&spd=100&seed=27930&text=lexgramm&mysent=&level1=0&level2=1&level3=1&level4=2&type4=&flags2=&type3=&type2=&flags4=&flags1=&flags3=&mysize=&mysentsize=&simple=1&env=alpha&parent2=1&link4=on&link3=on&link2=on&gramm4=S&gramm1=S&gramm2=V%2C%D0%BF%D1%80%D0%B8%D1%87&gramm3=PR&min2=&min3=1&min4=1&lang=ru&lex4=&lex1=&lex3=&max2=&max3=&lex2=&mycorp=&max4=&notag=1&parent4=3&parent3=1&mode=syntax&parent1=0

Ссылка для запроса по схеме 2:

http://processing.ruscorpora.ru/search.xml?sort=i_grtagging&out=normal&dpp=100&spd=100&seed=9153&text=lexgramm&mysent=&level1=0&level2=1&level3=2&level4=3&type4=&flags2=&type3=&type2=&flags4=&flags1=&flags3=&mysize=&mysentsize=&simple=1&env=alpha&parent2=1&link4=on&link3=on&link2=on&gramm4=S&gramm1=S&gramm2=V%2C%D0%BF%D1%80%D0%B8%D1%87&gramm3=PR&min2=1&min3=1&min4=1&lang=ru&lex4=&lex1=&lex3=&max2=&max3=&lex2=&mycorp=&max4=&notag=1&parent4=3&parent3=2&mode=syntax&parent1=0

Подтип 1.5 : схемы

1. **прич**+прил+пр+сущ(рис. 9)
2. прич+**прил**+пр+сущ (рис. 10)

здесь конкурирующие хозяева прич и прил.

Пример:

(5) «объявленные вредными для экономики» [Я. Е. Григорьевич 2019]

Формы поиска для запроса 1.5 представлены на рисунках 9 и 10:

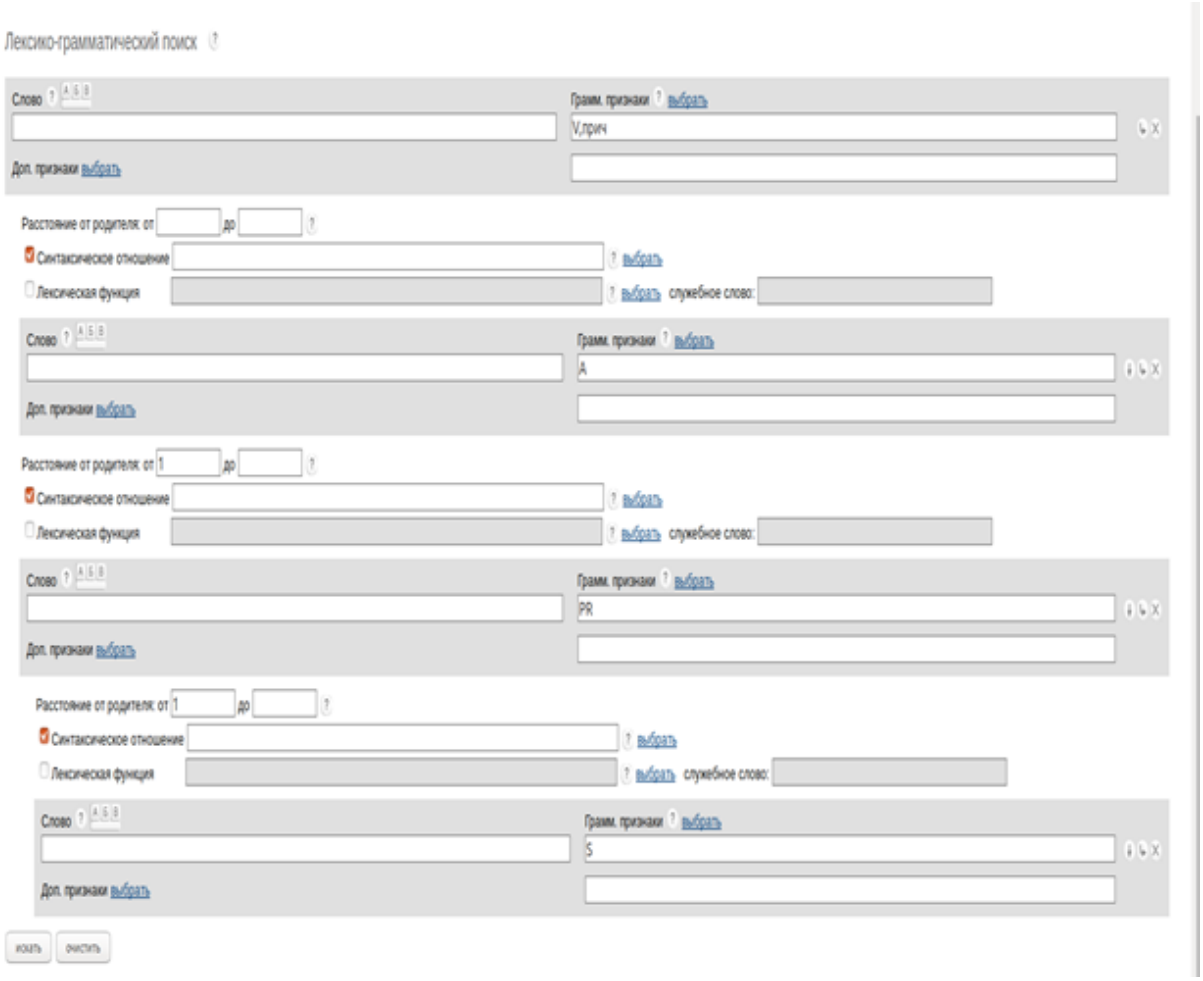


Рисунок 9. Форма запроса для схемы 1.

Сирконстант — предложная группа, которая зависит от глагола в форме причастия; линейный порядок элементов предложной группы задан с помощью расстояния — существительное находится на расстоянии 1 от хозяина, т.е. от предлога. От глагола в форме причастия также зависит прилагательное. Линейный порядок элементов задан с помощью расстояний: глагол в форме причастия — хозяин, на расстоянии 1 от него находится предложная группа, прилагательное может находиться между хозяином и предложной группой.

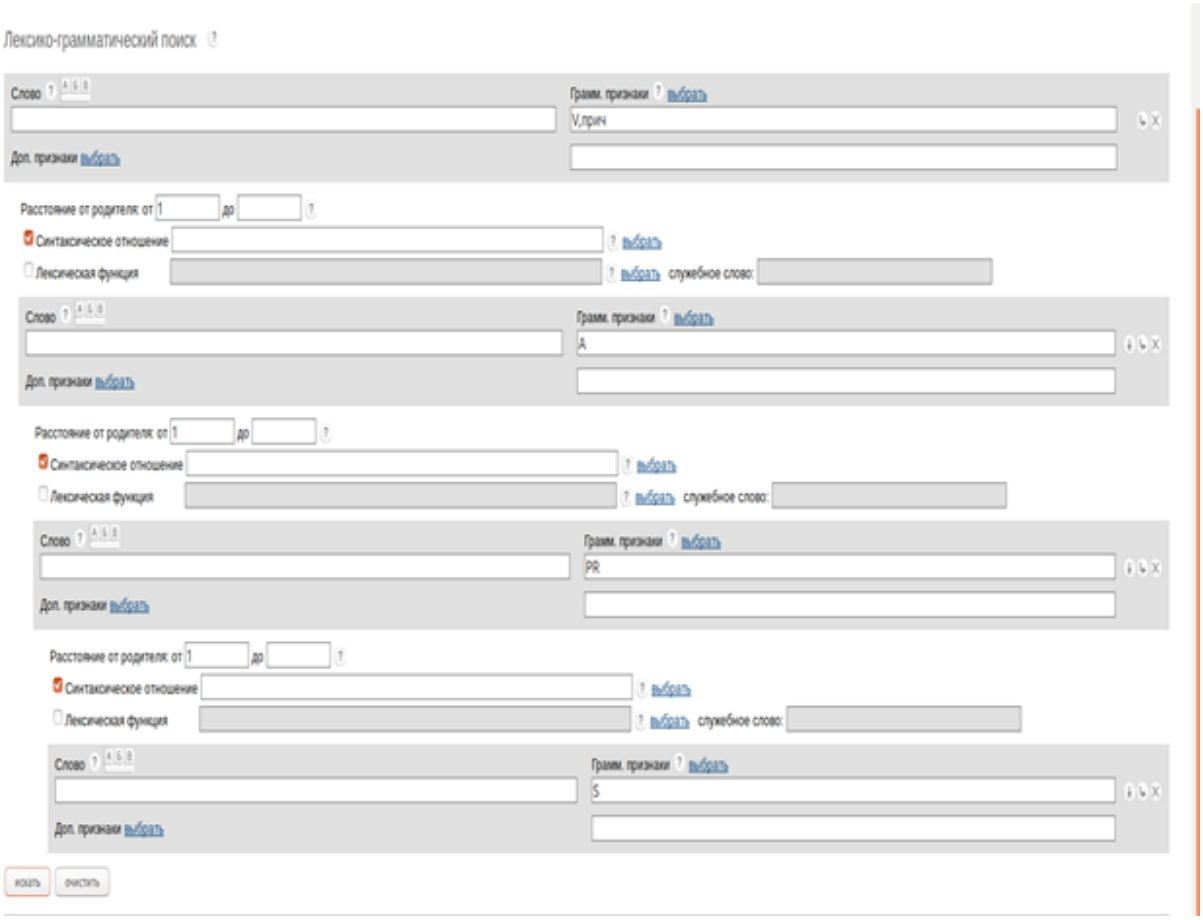


Рисунок 10. Форма запроса для схемы 2.

Сирконстант — предложная группа, которая зависит от прилагательного; линейный порядок элементов предложной группы задан с помощью расстояния — существительное находится на расстоянии 1 от хозяина, т.е. от предлога. Прилагательное зависит от глагола в форме причастия. Линейный порядок элементов задан с помощью расстояний: глагол в форме причастия — хозяин, на расстоянии 1 от него находится прилагательное, на расстоянии 1 от прилагательного находится предложная группа.

Представлены ссылки на запросы для подтипа 1.5:

Ссылка для запроса по схеме 1:

http://processing.ruscorpora.ru/search.xml?sort=i_grtagging&out=normal&dpp=100&spd=100&seed=4633&text=lexgramm&mysent=&level1=0&level2=1&level3=1&level4=2&type4=&flags2=&type3=&type2=&flags4=&flags1=&flags3=&mysize=&mysentsize=&simple=1&env=alpha&parent2=1&link4=on&link3=on&link2

=on&gramm4=S&gramm1=V%2C%D0%BF%D1%80%D0%B8%D1%87&gramm2=A&gramm3=PR&min2=&min3=1&min4=1&lang=ru&lex4=&lex1=&lex3=&max2=&max3=&lex2=&mycorp=&max4=&notag=1&parent4=3&parent3=1&mode=syntax&parent1=0

Ссылка для запроса по схеме 2:

http://processing.ruscorpora.ru/search.xml?sort=i_grtagging&out=normal&dpp=100&spd=100&seed=16007&text=lexgramm&mysent=&level1=0&level2=1&level3=2&level4=3&type4=&flags2=&type3=&type2=&flags4=&flags1=&flags3=&mysize=&mysentsize=&simple=1&env=alpha&parent2=1&link4=on&link3=on&link2=on&gramm4=S&gramm1=V%2C%D0%BF%D1%80%D0%B8%D1%87&gramm2=A&gramm3=PR&min2=1&min3=1&min4=1&lang=ru&lex4=&lex1=&lex3=&max2=&max3=&lex2=&mycorp=&max4=&notag=1&parent4=3&parent3=2&mode=syntax&parent1=0

2. Сирконстант — это **наречие**, у которого также может быть один из двух хозяев: гл или прич — это конструкция типа 2 по А.В. Гладкому (см. параграф 1.3.1).

Пример:

(6)«содержался сделанный наконец выбор» [М. Ходорковский 2019]

Схемы:

1.**гл** + нар + прич + сущ(рис. 11)

2.гл + нар + **прич** + сущ(рис. 12)

Формы поиска для запроса 2 представлены на рисунках 11 и 12:

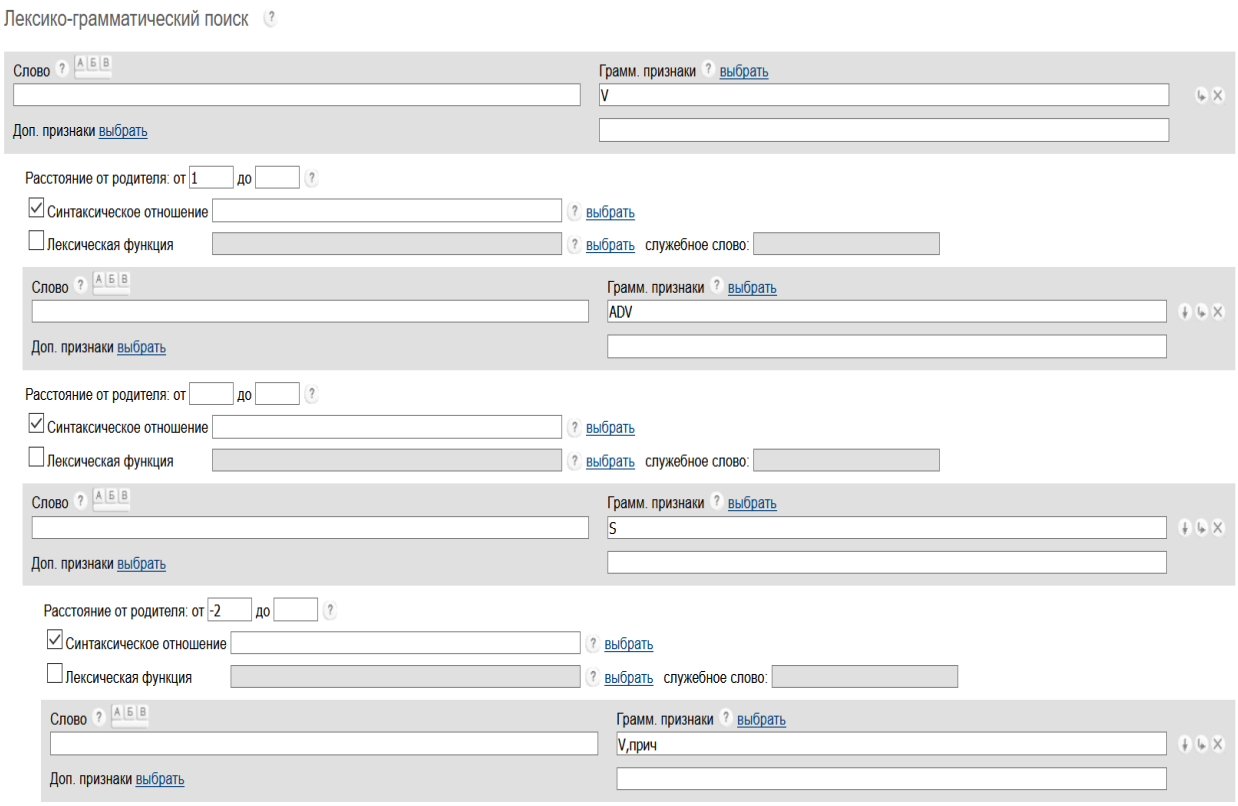


Рисунок 11. Форма запроса для схемы 1.

Сирконстант — наречие, которое зависит от глагола — хозяина и находится на расстоянии от 1, также от глагола зависит группа существительного с зависимым глаголом в форме причастия, которая находится сразу после наречия. Внутри группы линейный порядок задан расстоянием так, чтобы глагол в форме причастия зависел от существительного, но стоял сразу после глагола перед наречием, т.е. линейно находился левее существительного и наречия, поэтому расстояние 2 от существительного.

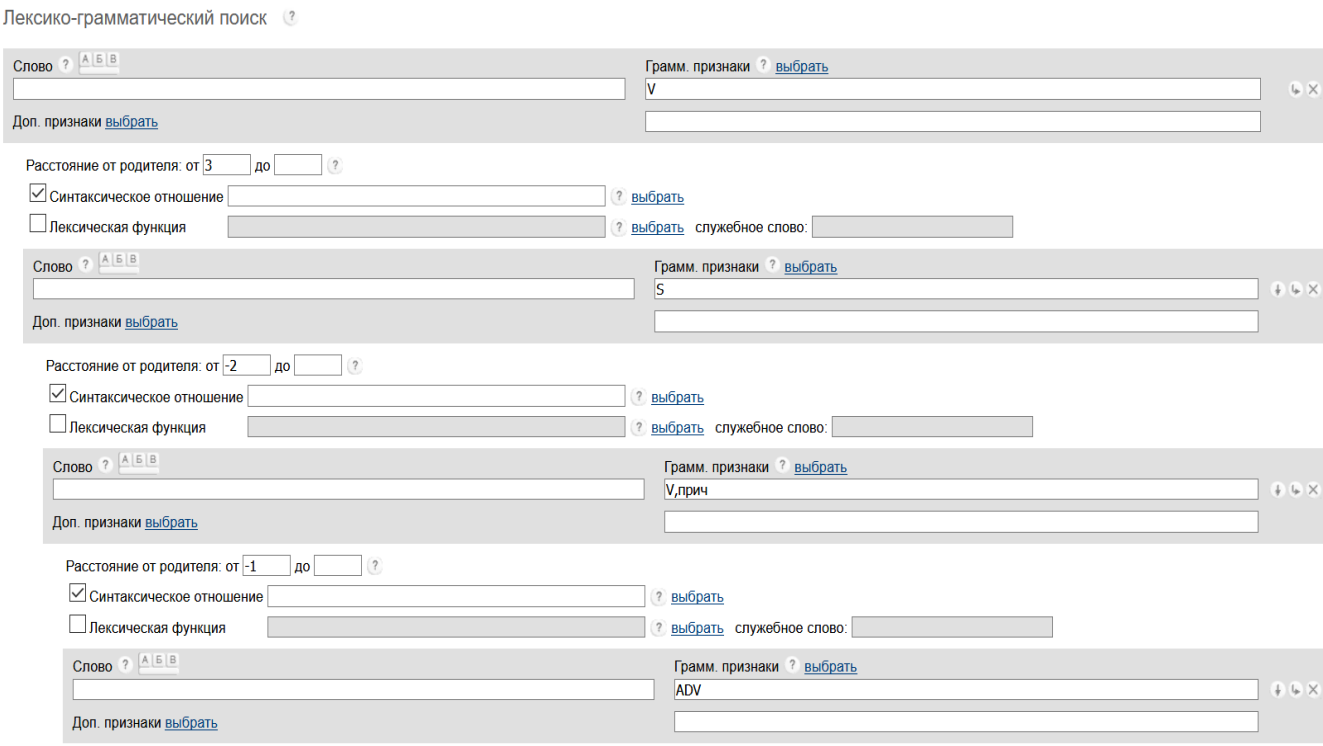


Рисунок 12. Форма запроса для схемы 2.

Сирконстант — наречие, которое зависит от глагола в форме причастия и находится на расстоянии от — 1(т.е. левее, чтобы соответствовать схеме), глагол в форме причастия зависит от существительного и линейно находится сразу после глагола — хозяина, те на расстоянии 2. Существительное зависит от глагола, но линейно находится после глагола в форме причастия и наречия, поэтому расстояние 3(т.е. правее остальных элементов).

Представлены ссылки на запросы для типа 2:

Ссылка для запроса по схеме 1:

http://processing.ruscorpora.ru/search.xml?sort=i_grtagging&out=normal&dpp=100&spd=100&seed=3357&simple=1&text=lexgramm&mysent=&level1=0&level2=1&level3=1&level4=2&type4=&type3=&type2=&flags4=&flags1=&flags3=&mysize=&mysentsize=&parent4=3&env=alpha&mode=syntax&link4=on&link3=on&link2=on&gramm4=V%2C%D0%BF%D1%80%D0%B8%D1%87&gramm1=V&gramm2=ADV&gramm3=S&lex2=&min2=1&min3=&min4=—2&lang=ru&lex4=&lex1=&lex3=&max2=&max3=&flags2=&mycorp=&max4=&notag=1&parent3=1&parent2=1&parent1=0

Ссылка для запроса по схеме 2:

http://processing.ruscorpora.ru/search.xml?sort=i_grtagging&out=normal&dpp=100&spd=100&seed=12064&text=lexgramm&mysent=&level1=0&level2=1&level3=2&level4=3&type4=&flags2=&type3=&type2=&flags4=&flags1=&flags3=&mysize=&mysentsize=&simple=1&env=alpha&parent2=1&link4=on&link3=on&link2=on&gramm4=ADV&gramm1=V&gramm2=S&gramm3=V%2C%D0%BF%D1%80%D0%B8%D1%87&min2=3&min3=—2&min4=—1&lang=ru&lex4=&lex1=&lex3=&max2=&max3=&lex2=&mycorp=&max4=&notag=1&parent4=3&parent3=2&mode=syntax&parent1=0

3. Сирконстант — **деепричастный оборот**, у которого также может быть один из двух хозяев: гл или инф — это конструкция типа 3 по А.В. Гладкому (см. параграф 1.3.1).

Пример:

(7)«может совпадать отличаясь» [Б.Руденко 2019]

Схемы:

1.**гл**+инф+дееприч(рис. 13)

2.гл+**инф**+дееприч(рис. 14)

Формы поиска для запроса 3 представлены на рисунках 13 и 14:

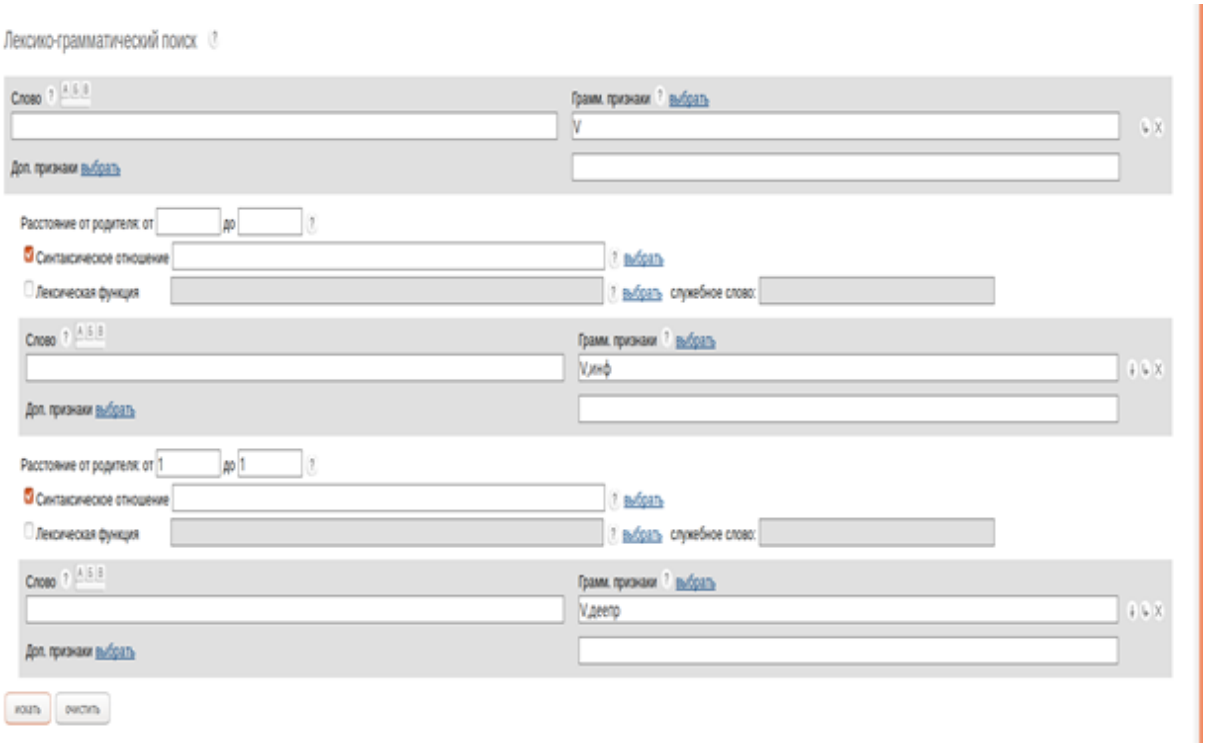


Рисунок 13. Форма запроса для схемы 1.

Сирконстант — деепричастие, которое также может быть выражено деепричастным оборотом; сирконстант зависит от глагола и находится на расстоянии от 1 от него. Также от глагола зависит глагол в форме инфинитива, который находится между глаголом — хозяином и сирконстантом.

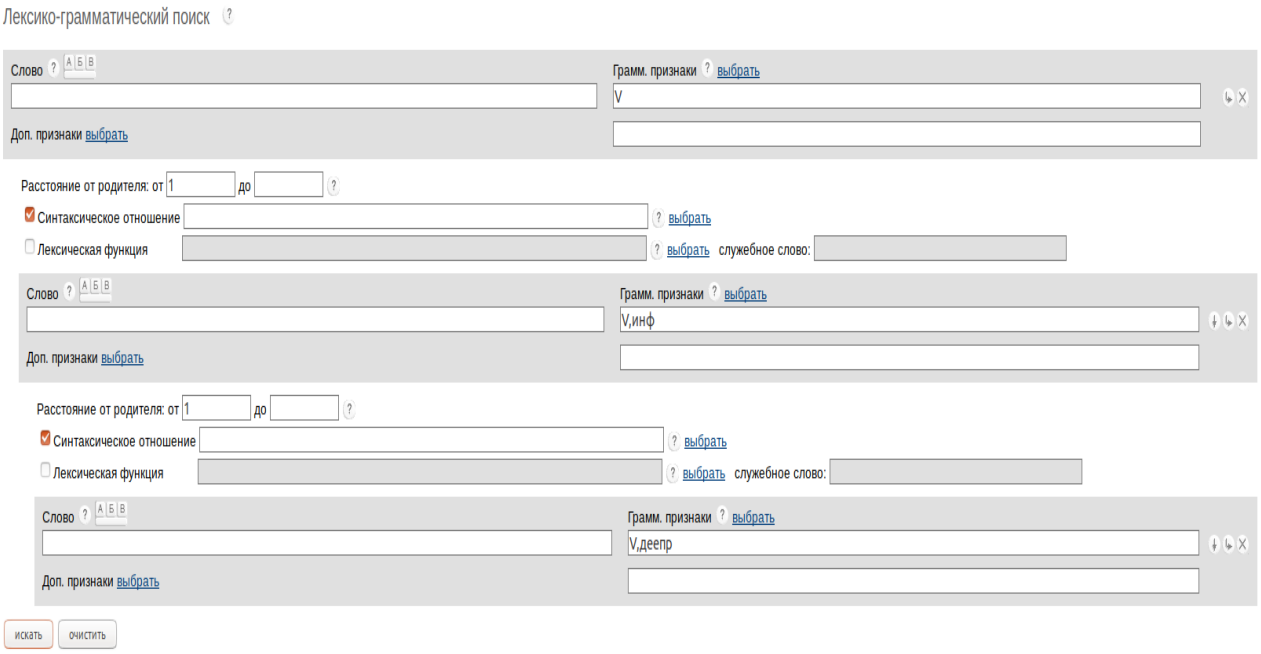


Рисунок 14. Форма запроса для схемы 2.

Сирконстант — деепричастие, которое также может быть выражено деепричастным оборотом; сирконстант зависит от глагола в форме инфинитива, который зависит от глагола — хозяина. Также от глагола зависит глагол в форме инфинитива, который находится между глаголом — хозяином и сирконстантом. Линейный порядок задан с помощью расстояний: элементы находятся на расстоянии от 1 друг от друга.

Представлены ссылки на запросы для типа 3:

Ссылка для запроса по схеме 1:

http://processing.ruscorpora.ru/search.xml?sort=i_grtagging&out=normal&dpp=100&spd=100&seed=19366&text=lexgramm&mysent=&level1=0&level2=1&level3=1&flags2=&type3=&type2=&flags1=&flags3=&mysize=&mysentsize=&simple=1&env=alpha&parent2=1&link3=on&link2=on&gramm1=V&gramm2=V%2C%D0%B8%D0%BD%D1%84&gramm3=V%2C%D0%B4%D0%B5%D0%B5%D0

%BF%D1%80&min2=&min3=1&lang=ru&lex1=&lex3=&max2=&max3=1&lex2=&mycorp=&notag=1&parent3=1&mode=syntax&parent1=0

Ссылка для запроса по схеме 2:

http://processing.ruscorpora.ru/search.xml?sort=i_grtagging&out=normal&dpp=100&spd=100&seed=31827&text=lexgramm&mysent=&level1=0&level2=1&level3=2&flags2=&type3=&type2=&flags1=&flags3=&mysize=&mysentsize=&simple=1&env=alpha&parent2=1&link3=on&link2=on&gramm1=V&gramm2=V%2C%D0%B8%D0%BD%D1%84&gramm3=V%2C%D0%B4%D0%B5%D0%B5%D0%BF%D1%80&min2=&min3=&lang=ru&lex1=&lex3=&max2=&max3=&lex2=&mycorp=&notag=1&parent3=2&mode=syntax&parent1=0

3.1.2 Алгоритм автоматической выгрузки неоднозначных конструкций в из синтаксического подкорпуса НКРЯ

Далее после составления запросов необходимо выполнить поиск результатов по каждому запросу и сформировать из всех найденных неоднозначных конструкций со стрелочной омонимией свой корпус для дальней шей обработки.

Результаты поиска выгружались автоматически: был написан алгоритм, который принимает на вход URL-ссылку на результаты поиска, осуществляет парсинг каждой страницы выдачи и в процессе парсинга ищет словоформы, соответствующие элементам конструкции. При этом изначально известно, сколько позиций должно быть заполнено в рамках одной конструкции. Алгоритм осуществлял поиск нужных элементов конструкции с помощью языка поиска по html документам xpath: в каждом предложении на странице выдачи находил подсвеченные слова — искомые элементы конструкции — и формировал из них строку из n нужных элементов, соответствующим количеству таких слов в пределах предложения. Так же для каждого слова находились его грамматические характеристики, из которых выделялась лемма. Из лемм конструкции

формировался список. Результат записывался в словарь, где конструкция — ключ, а список лемм — значение. Готовый словарь со всеми парами конструкция — список лемм записывался в csv файл.

Далее данные из файла были загружены в корпус лингвистического процессора AIIRE. Программный код алгоритма на языке Python3 представлен в Приложении 1. В ходе работы над алгоритмом возникали следующие проблемы:

1. долгое время ожидания результатов поиска, которое значительно увеличивало время работы программы;
2. периодическое отсутствие соединения с сервером (ошибка gateway timeout), что приводило к сбою работы алгоритма;
3. ошибка доступа AssertionError и системная ошибка OSError, которые так же приводили к прекращению работы алгоритма;
4. переход сайта на новую версию, подразумевающий изменения в структуре HTML и использование других кодировок, так же затруднял обработку;
5. ссылки на запросы были составлены до начала выгрузки и после перехода корпуса на новую версию стали недействительными.

Медленную работу поиска можно объяснить тем, что процент конструкций со стрелочной омонимией в НКРЯ крайне велик.

Для решения возникших вопросов было сделано следующее:

1. Время ожидания обработки каждой страницы запроса было увеличено с помощью добавления временной паузы в 3 секунды. На случай возникновения ошибки была предусмотрена пауза в 8 секунд. Также количество страниц, которые необходимо обработать было сокращено. С помощью корректировки адреса запроса: были увеличены значения параметров dpp (document per page), spp (sentence per page) до максимально возможных для относительно быстрой обработки;
2. Каждая страница с результатами обрабатывалась отдельно;

3. Был добавлен блок кода, содержащий команды для обработки ошибки AssertionError и отдельный блок для обработки OSError;
4. Изменения в структуре HTML и кодировки были учтены в коде;
5. Запросы были составлены заново уже для новой версии корпуса.

Пример результата работы алгоритма – содержимое txt файла с конструкциями – представлен в Приложении 2.

После завершения поиска и выгрузки неоднозначных конструкций необходимо проверить, пересекаются ли результаты поиска по двум вариантам запросов для одной конструкции, чтобы понять, находятся ли одни и те же конструкции и, соответственно, являются ли они неоднозначными в корпусе (те существуют ли для них две интерпретации или же они трактуются однозначно).

3.2 Алгоритм для получения статистических данных по выбранным из корпуса конструкциям

Алгоритм составления частотных словарей был следующим: в процессе парсинга html-страницы для каждого элемента конструкции с помощью языка поиска элементов по html xpath [Электронный ресурс] XPath//URL:https://ru.wikipedia.org/wiki/XPath (дата обращения: 05.11.19) выделялись заранее указанные в разметке страницы лемма и грамматические характеристики. Далее в грамматических характеристиках каждой словоформы находился частеречный тэг, в соответствии с которым лемма добавлялась в один из соответствующих словарей в качестве ключа; значение по ключу – частота встречаемости словоформы в формируемом корпусе, которая рассчитывалась автоматически в ходе работы алгоритма.

После завершения работы словари записывались в файлы в формате CSV.

Для каждого запроса формировались отдельные частотные словари. Число словарей определялось количеством разных частей речи в пределах одного запроса.

Пример полученного частотного словаря представлен в Приложении 3.

3.2.1 Построение выборки лексических единиц для компьютерного моделирования лексических значений

3.2.1.1 Объединение частотных словарей

Прежде чем начать построение выборки лексических единиц, необходимо было объединить ранее полученные частотные словари. Эта необходимость объясняется тем, что по каждому запросу был получен свой набор частотных словарей, а для выбора единого списка конструкций по всем типам конструкций и, соответственно, запросам, необходимо установить общую частотность каждой лексической единицы.

Алгоритм объединения был следующим:

1. Для каждой части речи выбрать все полученные для нее частотные словари и сохранить соответствующие файлы в отдельные директории;
2. Считывать файлы из директории по очереди и преобразовать в экземпляр класса dict(словарь);
3. В каждом полученном словаре перебрать все пары ключей и значений, то есть все пары лемм и их частот для каждой такой пары;
4. Если леммы нет в словаре, то добавить в новый пустой словарь лемму и ее частоту;
5. Если лемма уже была добавлена в словарь, то суммировать предыдущую частоту с новой;
6. Отсортировать полученный словарь и записать его в результирующий файл.

В результате для каждой из рассмотренных частей речи: глагол, существительное, предлог, наречие, прилагательное, — был получен объединенный частотный словарь. Далее на основе данных из объединенных частотных словарей была построена выборка лексических единиц.

Реализация алгоритма объединения частотных словарей представлена в Приложении 4.

3.2.2 Алгоритм построения выборки конструкций на основе выбранных лексических единиц

Для построения выборки лексических единиц был разработан следующий алгоритм, который заключался в том, чтобы найти среди ранее выгруженных из корпуса НКРЯ конструкций те, в которые входят только самые частотные леммы:

1. Был написан генератор лемм, который берет на вход список, составными элементами которого являются списки лемм разных частей речи. Каждый элемент списка лемм представляет собой тип данных кортеж, в котором содержатся в виде строк лемма, ее частота и ее частеречный тег. Элементы отсортированы по убыванию частот лемм:
   1.1. В процессе работы генератор проходит сразу по всем спискам лемм. На каждом шаге он берет i-ые элементы всех списков, находит лемму с наибольшей частотой и возвращает ее с помощью команды yield;
2. Входные данные для генератора были получены следующим образом: был написан алгоритм, который последовательно перебирает данные из всех файлов с частотными списками лемм для каждой части речи. В каждом таком списке осуществляется поиск леммы, ее частоты и частеречного тега. Данная информация записывается в тип данных кортеж. Полученный кортеж добавляется в соответствующий список кортежей для настоящей части речи; в конце все такие списки добавляются в результирующий список;
3. Далее был написан блок кода, в котором формировался конечный список конструкций с самыми частотными леммами:
   3.1. были посчитаны суммарные частоты лемм для каждой части речи: 398 646 для существительных, 70 482 для глаголов, 47 305 для предлогов, 82 для наречий и 47 для прилагательных. Леммы, частота которых меньше 2, не учитывались и были удалены из списков лемм;

3.2. на основе этих частот была посчитана суммарная частота для всех лемм, которая равна 516 562;

3.3. далее был запущен цикл по генератору лемм, где на каждом шаге полученная лемма добавлялась в список уже пройденных лемм (при условии, что ранее она туда не входила); после перебирались все конструкции из общего файла со всеми конструкциями. Этот файл был получен с помощью отдельного алгоритма, который показан в Приложении 5; его суть в объединении содержимого всех файлов с конструкциями по всем запросам в один общий файл. Те конструкции, в которых содержались все леммы из списка, записывались в результирующий файл; цикл повторялся до тех пор, пока сумма частот генерируемых лемм, деленная на суммарную частоту всех лемм не превышала 80% в соответствие с законом Парето и критериями, описанными в параграфе 2.4;

3.4. В конце был получен результирующий файл формата txt, содержащий все необходимые конструкции.

Общее число выгруженных из НКРЯ конструкций составило 22 703.

В результате работы данного алгоритма была получена выборка 8 592 конструкций, что составляет 38% от общего числа.

Так же был получен отдельный файл, содержащий леммы из конструкций в количестве 2 206 — именно столько понятий в онтологии потребуется обработать.

Далее эти конструкции были загружены в корпус-менеджер AIIRE.

Реализация алгоритма построения выборки конструкций представлена в Приложении 6. Пример некоторых конструкций из полученной выборки представлен в Приложении 7.

3.2.3 Загрузка конструкций в корпус-менеджер

Для автоматической загрузки неоднозначных конструкций в корпус-менеджер был использован следующий алгоритм:

1. Текстовый файл с неоднозначными конструкциями был преобразован в SQL - скрипт, в котором для каждой конструкции была вызвана SQL- команда INSERT;
2. Скрипт запускался администратором корпус-менеджера;
3. Во время работы скрипта команда INSERT выполнялась для каждой конструкции и они последовательно добавлялись в базу данных корпус-менеджера;
4. В итоге был получен корпус неоднозначных конструкций;
5. Корпус представляет собой список документов, в каждом из которых хранится одна конструкция;
6. Одна из конструкций была удалена («принять его за с»), так как оказалась неосмысленной;
7. Для того, чтобы увидеть результат парсинга, необходимо «процессировать корпус», то есть осуществить последовательный автоматический разбор каждой конструкции. «Процессирование корпуса» делается после того, как будет завершено компьютерное моделирование значений.

3.3 Компьютерное моделирование значений лексических единиц в составе неоднозначных конструкций.

Моделирование значений лексических единиц подразумевало описание в онтологии всех понятий, которые соответствуют леммам из полученного частотного списка лемм.

Примеры моделирования существительного:

Для моделирования существительного «*карточка*» были выполнены следующие действия:

1. Соответствующее существительному понятие было найдено в онтологии;
2. Было проверено по контекстам из выборки, в каком значении или значениях существительное употреблено. Контексты были

следующие: «*вам дают по карточкам продукты*», «*вводят карточки на продукты*», «*замена карточек на карты*». Значения, соответствующие контекстам были следующие:

a. «*талон для получения продуктов*»,

b. «*средство для идентификации личности*»

Значение b уже присутствовало в онтологии, к нему в качестве гиперонима было добавлено существительное «*идентификатор*» в значении «*объект, используемый для отождествления объектов опознания, их классификации*». Гипероним корректен. Имеющиеся синонимы «*карта*», «*пластиковая карта*», «*пластиковая карточка*» так же корректны. Соответствующий глагол у данного существительного отсутствует, добавлять атрибуты или генитивное отношение необходимости не было. Таким образом, обрабатывать значение b не пришлось.

Значения a в онтологии не было, оно было создано. Формулировка значения была взята из Викисловаря: «*истор. талон на получение определенного количества товаров*» [Электронный ресурс] Викисловарь// URL:https://ru.wiktionary.org/wiki/%D0%BA%D0%B0%D1%80%D1%82%D0%BE%D1%87%D0%BA%D0%B0 (дата обращения 24.05.20). Были добавлены синоним «*талон*» и гипероним «*документ*» в значении «*свидетельство чего-либо*». Гипероним и синоним корректны и обработаны в соответствие с конвенцией. Скриншоты с результатами моделирования представлены в Приложении 8 в качестве примера работы в онтологии AIIRE.

Трудности при моделировании существительных были следующие:

некоторые значения пересекались по базовым классам и таким образом становились неразличимыми. Например, такая трудность возникла при моделировании понятия «*робот*» с контекстами «*парк роботов в США*», «*создать робота с системой*», «*группы роботов с возможностями*», «*роботов действующих в среде*» и так далее. Для понятия «*робот*» было нужно значение «*техн. электромеханическое, пневматическое, гидравлическое устройство или их комбинация, предназначенное для замены*

человека в промышленности, опасных средах и др» с гиперонимом «*устройство*» в значении «*искусственный инструмент, рукотворный объект со сложной внутренней структурой, созданный для выполнения определенных функций*». Данное значение уже присутствовало в онтологии, однако пересекалось со значением «*боевой робот — устройство автоматики, заменяющие человека в боевых ситуациях или для работы в условиях, несовместимых с возможностями человека*». Из-за получавшегося пересечения понятия были неразличимы на уровне базовых классов, что могло в последствие привести к некорректному разбору конструкций со словом «*робот*». Проблема была решена следующим образом: значение «*боевой робот*» было оформлено как отдельное понятие, которое является гипонимом понятия «*робот*» и так же является типичным представителем класса устройств-роботов.

Пример моделирования глагола:

Было необходимо выполнить моделирование глагола «*лежать*» в значении «*иметься где-либо; содержаться, заключаться в чем-либо*» [Электронный ресурс] Викисоварь//URL:https://ru.wiktionary.org/wiki/%D0%BB%D0%B5%D0%B6%D0%B0%D1%82%D1%8C (дата обращения: 03.02.20) с контекстами «*идея лежит в основе*», «*Она лежит в основе*», «*Решение лежит на поверхности*», «*механизм лежит в основе*». Было добавлено новое значение, выбран тип «*состояние*», указан технические процесс, субъект — «*понятие*», как общий базовый класс понятий «*идея*», «*решение*», «*механизм*»; действие ненаправленное и неадресованное.

А для контекста «*ответственность лежит на руководстве*» было добавлено значение «*составлять чью-либо обязанность, долг, занятие и т. п.*» [там же] тип «состояние», технический процесс и субъект «атрибут» как вышестоящее понятие по отношению к «*ответственности*». Действие так же ненаправленное и неадресованное. Результат представлен в приложении 9 в качестве примера работы в Ontohelper.

Пример моделирования прилагательного:

Алгоритм моделирования прилагательного «*значимый*» в значении «*несущий важное значение*»:

1. прилагательное значимый обозначает свойство «*роль*», некая сущность обладает данным свойством
2. такой сущностью может быть «*объект, свойство или процесс*», то есть буквально быть значимым может что угодно
3. существительное «*роль*» присутствует в онтологии со значением «*значение звена в системе отношений*», гиперонимом является понятие «*свойство*» в значении «*отношение при концепте, характеризующее его*»; синонимы «*назначение*» и «*функция*» корректны, генитивные отношения не требуются
4. необходимое отношение для «*роль*» «*принадлежать объекту, свойству или процессу*», а для понятия «*объект, свойство или процесс*» необходимо, соответственно, обратное отношение «*обладать ролью*»
5. «*роль*» должна быть «*значимой*», то есть необходимо отдельное понятие «значимая роль», являющееся гипонимом «*роли*»
6. «*значимая роль*» должна быть атрибутом «*объекта, свойства или процесса*», посредством установленного отношения она им является
7. прилагательное «*значимый*» является переменный атрибутом, так как существует существительное на -ость «*значимость*»

Пример моделирование предлога:

Алгоритм моделирование предлога «*в*» в значении *«(о воре) действовать в рамках своего собственного закона*» для специфической конструкции «*вор в законе*»:

1. добавлено новое значение предлогу «*в*»

2. добавлен гипероним «*бинарное отношение*» в значении «*отношение между двумя сущностями*»

Пример моделирования местоимения:

Алгоритм моделирования местоимения «*он*»:

1. добавлено значение «*местоимение третьего лица (лемма для местоимений)*»
2. добавлен гипероним «*понятие*» в значении «*отображенное в мышлении единство существенных свойств, связей и отношений предметов или явлений*»

Наречия не были рассмотрены, так как они не вошли в выборку.

3.4 Компьютерное моделирование семантических валентностей для разрешения синтаксической неоднозначности в онтологии AIIRE.

После устранения разрывов производится парсинг и получаются синтаксические деревья и соответствующие им семантические графы. В деревьях прописываются синтаксические связи между словами, а в графах — соответствующие семантические отношения между концептами. Вариантов разбора может быть довольно большое количество. Чтобы понять, какие версии корректны, а какие — ошибочны, их нужно сравнить между собой.

Самый частый случай возникновения нескольких вариантов разбора — это наличие нескольких значений у одной или более лексических единиц. Тогда нужно проверить, те ли значения представлены в семантическом графе. После выбора нужных значений для данной конструкции, нужно также проверить отношения между ее элементами, которые тоже могут быть разными. «Ненужные» и не подходящие по смыслу отношения уточняются в соответствии с логикой и языковым материалом путем повторного моделирования концептов в онтологии и ontohelper, где устанавливаются

только корректные отношения. Например, в конструкции *«участие иудеев в делах»* получилось 4 варианта разбора. В двух из них лексические единицы участие и иудеев находились в генетивном отношении: «*иудеи*» обладали свойством «*участие*», а не принимали участие в чем-либо. Чтобы устранить эту ошибку, нужно было зайти в онтологию, выбрать нужное значение концепта участие: «*вовлеченность в какую-либо деятельность*», добавить отношение «*соответствие процесса действию*» и указать объект данного отношения «*участвовать*». После было доработано понятие «*иудеи*»: добавлен гипероним «*последователь*», который, в свою очередь, связан с базовым понятием «*некто*». Последний понадобился при моделировании глагола участвовать в ontohelper, где были выбраны правильный субъект («*некто*») и предлог в нужном значении *«(об участии) относиться к объекту или процессу»*. В конце, после повторного моделирования были получены только два корректных варианта участие иудеев в делах, где «иудеи» принимают участие в процессе.

Еще один пример: конструкция «*представление схемы в виде логических элементов*». В результате парсинга, в одной из версий «*представление*» находится в отношении принадлежности к «*схеме*». Чтобы понять, что это за отношение, нужно было проверить семантические графы, соответствующие данной версии синтаксического разбора. В семантическом графе видно, что «*схема*» в значении «*структура*» принадлежит совокупности «*представление*» в значении «*воображение*». Это происходит потому, что воображение — это процесс, а процесс — совокупность событий, но проблема в том, что структура — не событие и потому не может быть элементом совокупности событий. Чтобы это исправить, нужно было добавить в онтологию к понятию совокупность событий отношение быть совокупностью, к которой относятся объекты или процессы с объектом ситуация. После добавления некорректный вариант разбора был устранен.

3.5 Оценка эффективности разрешения неоднозначности

В результате проделанной работы по определению возможностей онтологической семантики в разрешении стрелочной омонимии в конструкциях с сирконстантами удалось добиться корректного разбора и устранения разрывов в 972 конструкциях из 8 591, что составляет 11,3% от всей выборки. Для получения такого результата было проверено и отредактировано и смоделировано около 3 151 понятия в онтологии AIIRE из общего числа понятий 82 841, в число которых входили как и 2 206 понятий из списка лемм, так и смежные с ними понятия, которые так же потребовалось уточнить для получения корректных результатов. Следует отметить, что из полученного списка лемм в рамках отведенного для выполнения данного исследования времени удалось проверить и отредактировать только 1 416 лемм, что несомненно повлияло на конечный результат. Основной причиной ошибочных разборов являются неустраненные разрывы в оставшихся 7 618 конструкциях, которые как раз являются результатом того, что часть понятий не была отредактирована или же была отредактирована не корректно. Следствием этого так же является возникновение ошибки ExplosionError в 811 конструкциях, во многих из которых данная ошибка была зафиксирована более одного раза в разных частях конструкции при формировании разных связей.

Информация о количестве разрывов и числе возникновения ошибки ExplosionError (ошибка комбинаторного взрыва) была взята из соответствующих файлов с отчетами об ошибках tree_gaps.csv и errors.csv, которые были автоматически созданы в ходе выполнения автоматического разрешения неоднозначности и далее скачаны из корпус-менеджера.

С учетом всего изложенного выше, можно дать следующую оценку:

Во-первых, следует отметить, что метод онтологической семантики может быть эффективным в решении задачи снятия синтаксической неоднозначности с сирконстантами, так как в 972 из 972 были получены корректные разборы, отсутствуют разрывы и ошибка ExplosionError. Пример

одного из таких разборов представлен в Приложении 10. Эффективность данного метода зависит от качества и правильности моделирования понятий в онтологии.

Во-вторых, надо сказать, что метод является трудоемким и его реализация требует значительного количества времени. Средняя скорость моделирования понятий в онтологии с учетом смежных с ними понятий составляет 10-20 понятий в день, при условии, что моделирование корректно.

Выводы к главе 3.

В данной главе были описаны практические действия и алгоритмы реализации поставленной задачи разрешения стрелочной омонимий методом онтологической семантики: формирование поисковых запросов на основе выделенных ранее (см главу 1) типов неоднозначных конструкций, алгоритм автоматической выгрузки результатов и сопутствующих ей технических проблем с их последующими решениями; формирования репрезентативной выборки конструкций на основе частотного анализа лемм в соответствие с законом Парето, загрузки конструкций в корпус-менеджер AIIRE и их последующее моделирование в онтологии, то есть моделирование их значений и семантических валентностей. Данные процедуры были продемонстрированы на конкретных примерах из получившейся выборки. В конце были приведены результаты и дана оценка эффективности используемого в работе метода.

Заключение

В процессе выполнения данной дипломной работы было проведено экспериментальное исследование возможностей онтологической семантики в

разрешении стрелочной омонимии в конструкциях с сирконстантами на материале синтаксически размеченного корпуса текстов на русском языке и была дана оценка трудоемкости и эффективности данного метода. Поставленные цели были достигнуты путем последовательного выполнения ряда необходимых задач:

1. была сформирована репрезентативная выборка употреблений русскоязычных конструкций с сирконстантами, характеризующихся стрелочной омонимией, необходимого и достаточного по содержанию и объему для исследования методов и оценки качества их работы
    1.1. были выделены типы и подтипы неоднозначных конструкций со стрелочной омонимией с сирконстантами, составлены схемы и на основе них сформулированы поисковые запросы к синтаксическому подкорпусу НКРЯ, в котором осуществлялся поиск;
    1.2. были разработаны средства автоматической выгрузки синтаксически неоднозначных конструкций, которые представляют собой алгоритм на языке Python
    1.3. был разработан алгоритм на языке Python, который строит частотные словари лемм каждой части речи из конструкций по каждому запросу и составляет выборки конструкций всех типов, содержащих наиболее частотные леммы, необходимой и достаточной по объему для обеспечения статистической достоверности выводов об исследуемых показателях эффективности автоматического разрешения неоднозначности и выдает таким образом репрезентативную выборку достаточного и необходимого объема
2. была осуществлена загрузка созданного репрезентативного корпуса конструкций в корпус-менеджер, обеспечивающий возможность автоматической синтаксической и семантической разметки корпуса при помощи лингвистического процессора, выполняющего разрешение

синтаксической неоднозначности средствами онтологической семантики.

3. было выполнено моделирование понятий, соответствующих значениям лексических единиц, употребляемых в корпусе, в онтологии, с учетом их семантических валентностей, ограничивающих возможности синтаксической интерпретации неоднозначных конструкций; обеспечение корректности версий автоматической синтаксической разметки конструкций лингвистическим процессором путем корректировки и задания семантических отношений на базовых классах концептов онтологии.
4. в конце были приведены анализ и оценка полученных результатов.

Так же была обоснована актуальность исследования путей решения поставленной проблемы синтаксической неоднозначности в конструкциях с сирконстантами, было дано теоретическое описание исследуемой проблемы и выбранного метода ее решения так же, как и обзор других методов; были описаны вспомогательные методы для решения практических задач, был дан обзор различных ресурсов и обоснован выбор лингвистического процессора AIIRE, на базе которого было проведено исследование.

Литература

1. Чернова Д.А. моделирование синтаксически неоднозначных предложений: психолингвистическое исследование — Дисс. ... кандидат филологические науки — Санкт-Петербург: Санкт-Петербургский государственный университет, 2016
2. Федорова О.В., Янович И.С. Разрешение синтаксической неоднозначности в русском языке: роль длины и структуры подчиненного. Диалог—2005. E.V.

3. Шкурко Е. В. Синтаксическая омонимия и способы предупреждения ее возникновения. — Днепропетровский национальный университет 2007, УДК 811.161.1'367.332
4. Yusuke Miyao, Jun'ichi Tsujii «A model of syntactic disambiguation based on lexicalized grammars«. Department of Computer Science, University of Tokyo 2003
5. Khalil Sima'an «ON MAXIMIZING METRICS FOR SYNTACTIC DIS-AMBIGUATION». Language and Inference Technology Group Institute for Logic, Language and Computation (ILLC) University of Amsterdam, The Netherlands 2003
6. Jakub Zavrel, Walter Daelemans, Jorn Veenstra «Resolving PP attachment Ambiguities with Memory-Based Learning». Computational Linguistics, Tilburg University PO Box 90153, 5000 LE Tilburg, The Netherlands 1997
7. А. Добров. Коллективная монография «Прикладная и компьютерная лингвистика». ЛЕНАНД. 2016. 35-58.
8. Daniel Jurafsky A. Probabilistic Model of Lexical and Syntactic Access and Disambiguation. Cognitive science, 1996, 20 137-194
9. Richard Johansson, Pierre Nugue Dependency-based syntactic-semantic analysis with PropBank and NomBank. Proceeding CoNLL 08 Proceedings of the Twelfth Conference on Computational Natural Language Learning 183-187, 2008
10. Наталья Гаранина, Елена Сидорова Контекстно—зависимая лексико-синтаксическая неоднозначность в популяции идеологов Институт информатики им. А.П. Ершова, пр. Лаврентьева, 6, Новосибирск 630090, Россия, 2016
11. Philipp Cimiano, Uwe Reyle Ontology-based semantic construction, under-specification and disambiguation 2003
12. Alexander F. Gelbukh Lexical, Syntactic, and Referencial Disambiguation Using a Semantic Network Dictionary Natural Language Processing Labora-

tory, Centro de Investigación en Computación, Instituto Politécnico Nacional. 07738 México D.F. 1998

13. Tatu J Ylonen Joint disambiguation of syntactic and semantic ambiguity Clausal Computing Oy, Helsinki (FI) 2011
14. А. Добров. Автоматическая рубрикация новостных сообщений с помощью синтаксической семантики. Дисс. ... кандидат Филологические науки — СПб: СПбГУ, 2014
15. Окатьев В.В., Гергель В.П., Алексеев В.Е., Таланов В.А., Баркалов К.А., Скатов Д.С., Ерехинская Т.Н., Котов А.Е., Титова А.В., с. Отчет о выполнении НИОКР по теме: «Разработка пилотной версии системы синтаксического анализа русского языка» (инвентарный номер ВНТИЦ 02200803750) — М .: ВНТИЦ, 2008
16. Гладкий А.В. Синтаксические структуры естественного языка в автоматизированных системах связи — М .: Наука, 1985
17. Митренина О.В. Проблемы разбора неоднозначности. Дисс. ... кандидат Филологические науки — СПБ: СПбГУ, 2005
18. Добров А.В. Semantic and Ontological Relations in AIIRE Language Processor. Computational Models for Business and Engineering Domains. — ITHEA, Rzeszow-Sofia 2014
19. Леонтьева Н.Н. Автоматическое понимание текстов. Системы, модели, ресурсы — М.: Academia, 2006
20. Алпатов В. М. История лингвистических учений. Учебное пособие. 2-е изд., испр. — М.: «Языки русской культуры» , 1999.
21. Теньер Л. Основы структурного синтаксиса—М.: «Прогресс» , 1988
22. Ярцева В.Н. Лингвистический энциклопедический словарь — М.: «Советская энциклопедия» , 1990
23. Neches R., Fikes R.E., Finin T., Gruber T.R. Patil, R. Senator T.,Swartout W.R. Enabling technology for knowledge sharing // AI Magazine. 1991. Vol. 12, №3, pp. 16-36
24. Nirenburg S., Raskin V. Ontological Semantics. Cambridge, MA, 2004

25. Gruber T.R. A translation approach to portable ontology specifications // Knowledge Acquisition, 5 (2), 1993
26. Мельчук И.А. Толково-комбинаторный словарь. — Russian Language Journal, 1984,38:129/130,189-198
27. Леонтьева Н.Н. Леонтьева Н.Н. Автоматическое понимание текстов Системы модели ресурсов — Академия, 2006. — 153 с. ISBN 5-7695-1842-1
28. П. Л. Гроховский, А. В. Добров, А. Е. Доброва, Н. Л. Сомс Корпус-менеджер для морфосинтаксической разметки: опыт разработки корпуса тибетских грамматических сочинений — Труды международной конференции «Корпусная лингвистика–2017». — СПб.: Изд-во С.-Петерб. Ун-та, 2017. — 340 с.
29. Азарова И. В., Браславский П. И., Захаров В. П., Киселев Ю. А., Усталов Д. А., Хохлова М. В. Идентификация единиц тезаурусного описания при интеграции лексических ресурсов RussNet и YARN — Структурная и прикладная лингвистика: межвуз. сб. С83 Вып. 12: К 60-летию отделения прикладной, компьютерной и математической лингвистики СПбГУ / отв. ред. И. С. Николаев. — СПб.: Изд-во С.-Петерб. Ун-та, 2018. — 34 с.
30. Азарова И.В., Синопальникова А.А., Яворская М.В.Принципы построения WordNet-тезауруса RussNet — материалы конференции Диалог 2004.
31. Годгильева М.М. Корпусно-структурный анализ как инструмент полуавтоматического явления значений и семантических валентностей глаголов русского языка — ВКР СПБГУ Санкт-Петербург 2017.

Электронные ресурсы

1. НКРЯ: Национальный корпус русского языка 2019[Электронный ресурс]. В. Дубнов Последний председатель URL: http://www.ruscorpora.ru (дата обращения 02.07.2019)
2. НКРЯ: Национальный корпус русского языка 2019 [Электронный ресурс]. Н. Добрецов На перекрестках всех миров URL: http://www.ruscorpora.ru (дата обращения 02.07.2019)
3. НКРЯ: Национальный корпус русского языка 2019[Электронный ресурс]. Я. Е. Григорьевич Не в то ВТО URL: http://www.ruscorpora.ru (дата обращения 02.07.2019)
4. НКРЯ: Национальный корпус русского языка 2019[Электронный ресурс]. М. Ходорковский Левый поворот URL: http://www.ruscorpora.ru (дата обращения 02.07.2019)
5. НКРЯ: Национальный корпус русского языка 2019[Электронный ресурс]. Б. Руденко Все возрасты равны URL: http://www.ruscorpora.ru (дата обращения 02.07.2019)
6. Конвенции и методики по работе с онтологией AIIRE [Электронный ресурс] URL:http://ontology.aiire.org/static/conventions.html (дата обращения 05.09.2019)
7. Методика моделирования отношений в AIIRE [Электронный ресурс] URL:https://docs.google.com/document/d/1-77ZqkCT0MfDV8zq6HkpSYWwM0S_8lpsNYta3gk5IbM/edit?skip_itp2_check=true#heading=h.g9ko9rpyn5gc (дата обращения 06.09.2019)
8. RussNet [Электронный ресурс] URL:http://ct05647.tmweb.ru/russnet/дата (обращения: 06.05.2020)
9. WordNet [Электронный ресурс] URL:www.wordnet.ru (дата обращения: 06.05.2020) и URL :https://wordnet.princeton.edu/obtain (дата обращения: 06.05.2020)
10. YARN [Электронный ресурс] URL :https://russianword.net/ (дата обращения 06.05.20) URL:https://russianword.net/yarn.xml (дата обращения 06.05.20)

Приложение 1. Алгоритм автоматической выгрузки конструкций из синтаксического подкорпуса НКРЯ

```
# для кодировок
import codecs
# для работы с тьюплами
from collections import namedtuple
# определяет функции и классы, которые помогают открывать URL
from urllib.request import urlopen
# для работы с csv
import csv
# для сортировки словарей
from collections import OrderedDict
from lxml import html
# модуль для работы со временем
import time
# множество функций для работы с операционной системой
import os
# предоставляет классы для обработки времени и даты разными способами
import datetime
# для декодирования суффикса
from urllib.parse import unquote

if __name__ == '__main__':
 ''' # создаю файл для конструкций
        Constructions = open('Constructions.txt', 'w')
        # потом я его открою на дозапись и буду по очереди дозаписывать все
конструкции
        Constructions.close()'''
# это ссылка для поиска слова
word_search_link = 'http://processing.ruscorpora.ru/search-
ex-
plan.xml?env=alpha&mycorp=&mysent=&mysize=&mysentsize=&dpp=&spp=&
spd=&&mode=syntax&notag=1&simple=1&lang=ru&parent1=0&level1=0&lex1
```

```
=&gramm1=V&flags1=&parent2=1&level2=1&min2=&max2=&link2=on&type2
=&lex2=&gramm2=S&flags2=&parent3=2&level3=2&min3=&max3=&link3=on
&type3=&lex3=&gramm3=PR&flags3=&text=word-
info&requestid=1569187164845&language=ru&source='
# это тьюпл для хранения информации о слове и самого слова
Word = namedtuple(«Word»,  «word lemma part_of_speech grammar_structure«)
# глобальный словарь, играющий роль промежуточного хранилища
global_dict = {}
# пауза отдыха после ошибки
PAUSE_AFTER_FAILURE = 8
# максимально количество попыток что то сделать
MAX_RETRY = 3
# файл, куда пишется вся выдача, чтобы не перегрузить консольку или IDLE
log_file = open('ruscorpora.log', 'w')

def log(*args):
        '''
        This function writes information in file and also prints it on the screen
        @param *args: arguments the function takes; the number of arguments is
not limited
        '''
        log_file.write(' '.join(str(arg) for arg in args))
        log_file.write('\n')
        log_file.flush()
        '''print(*args)'''

def get_lemma_and_params(suff):
        # достаю лемму и грам параметры из suff
        '''log('I am in get lemma and params')'''
        # парсю ссылку на слово + suff
        parsed_url = html.parse(word_search_link + suff)
        # нахожу и запоминаю нужную информацию о слове, она там вся сразу
        # и существует в виде двух элементов леммы и параметров
        info = parsed_url.xpath('//td[@class=«value«]/text()')
        # если их точно два, вытаскиваю их по очереди, иначе - ошибка
        if len(info) < 2:
         try:
         log('Failed to find lemma and params, sorry:(')
         log(html.tostring(parsed_url))
```

```
        except Exception:
        raise
       else:
       # тащу и декодирую лемму
       lemma = bytes([ord(c) for c in info[0]]).decode('utf-8').replace(' ',
'').replace('\n(', '')
       # тащу и декодирую все грамматические характеристики
       params = bytes([ord(c) for c in info[1]]).decode('utf-8').replace(' ',
'').replace('\n(', '')
       log(params, 'params')
       return lemma, params
       def get_word_info(word, suff, s_freq_dict, pr_freq_dict, v_freq_dict,
a_freq_dict, adv_freq_dict):
       '''
       This function gets information about a given word and makes frequency dic-
tionaries
       for each part of speech (noun, preposition, verb)
       @param word: string containing a word
       @param suff: string containing search suffix to find a word
       @return: csv files with frequency dictionaries
       '''
       constr_str = ' '
       # начинаю создавать частотные словари
       try:
        # если suff отсутствует, добавляю в глобальный словарь
        if suff not in global_dict:
        lemma, params = get_lemma_and_params(suff)
        global_dict[suff] = (lemma, params)
        else:
        # иначе достаю его из этого словаря
        # это общий словарь
        lemma, params = global_dict[suff]
        # это все грам параметры слова со всеми характеристиками и еще
лемма
        pr_set = set(params.split(«, \xa0«) )
        # ищу нужный тег и по нему добавляю лемму в соответствующий
частотник
        if 's' in pr_set :
        s_freq_dict.setdefault(lemma, 0)
```

```
            s_freq_dict[lemma] += 1

            elif 'pr' in pr_set:
            pr_freq_dict.setdefault(lemma, 0)
            pr_freq_dict[lemma] += 1

            elif 'v' in pr_set:
            v_freq_dict.setdefault(lemma, 0)
            v_freq_dict[lemma] += 1

            elif 'adv'in pr_set:
            adv_freq_dict.setdefault(lemma, 0)
            adv_freq_dict[lemma] += 1

            elif 'a' in pr_set:
            a_freq_dict.setdefault(lemma, 0)
            a_freq_dict[lemma] += 1

            else:
            # на случай если тег не найден
            log('Error! Tag not found!')
            log(word)
            # поднимаю ошибку. здесь так же возникает OSError, я ее обнаружила,
когда качала запрос 5
 except OSError:
            log('BAD OSError happened!')
            except AssertionError:
            log('AssertionError')

def search_highlighted(url, s_freq_dict, pr_freq_dict, v_freq_dict, a_freq_dict,
constr_dict, adv_freq_dict):
            '''
            This function searches all highlighted words and makes constructions
            and forms constr_dict={'construction':[lemma0,lemma1,lemma2,lemma3]}
            @param url: string adress of a searching query
            '''
            log('search')
            # парсю ссылку
            parsed_url = html.parse(url)
```

```
        log('I parsed url')
        # создаю список конструкций и открываю файл на дозапись
        constr_str = []
        log(len(parsed_url.xpath('//div[@class=«content«]/ol/li/table/tr/td/ul/li')))
        # для каждого предложения из контента, те из документа на странице
начинаю искать выделенные слова
        for num, sent in enumerate
(parsed_url.xpath('//div[@class=«content«]/ol/li/table/tr/td/ul/li')):
         # пустая сторока куда буду плюсовать слова в конструкцию
         constr = ''
         # список всех лемм конструкции
         lemmas = []
         log(num, 'sent_num')
         # слежу за временем обработки предложения
         log(datetime.datetime.now().isoformat())
         # нахожу полное предложение
         full_sent = sent.xpath('normalize-space(.)').split(' [', 1)[0]
         # в нем нахожу выделенное слово
         for highlighted_word in sent.xpath('span[@class=«b-wrd-expl g-em«]'):
         # отдыхаю, чтобы не перетрудиться
         time.sleep(3)
         '''log('I got to the word!')'''
         # достаю слово в виде текста
         word = highlighted_word.xpath('text()')
         word = ''.join(word)
         # декодирую в ютф-8
         word = bytes([ord(c) for c in word]).decode('utf-8')
         '''log(word, «word_decoded«)'''
         # нахожу грамматические характеристики слова, они же suff
         suff = highlighted_word.xpath('@explain')
         # вытаскиваю суффикс
         lem_suff = suff[0] # %D0%B2%7C5%7C%D0%B2
         # декодирую суффикс
         lem_suff = unquote(lem_suff)
         '''print(type(lem_suff),'type')
         print(lem_suff,'suff')'''
         # сплит вернет список, где 1 элемент это моя лемма
         lemma = lem_suff.split('|')[0]
         log(lemma,«lemma«)
```

```
        lemmas.append(lemma)
        # если то, что я нашла не пустой результат поиска, передаю
        # грамматические характеристики и пустые словари в функцию поиска нужных характеристик(частеречных тегов)
        if len(word)==0 or len(suff)==0:
        log('No such word or suff')
        else:
        get_word_info(word[0], suff[0], s_freq_dict, pr_freq_dict, v_freq_dict, a_freq_dict, adv_freq_dict)
        # не забываю отдыхать
        time.sleep(3)
        # собираю слова в конструкцию
        constr+=word+' '
        log(constr, «CONSTR!«)
        log (lemmas, «Lemmas!«)
 # добавляю в словарь конструкцию и список ее лемм
        constr_dict.setdefault(constr, []).append(lemmas)
        '''# добавляю констрцукцию в результирующий файл
        Constructions.write(constr + '\n')'''
        '''# не забываю закрыть файл, чтобы потом снова открыть и дозаписать новую конструкцию без утраты старой
 Constructions.close()'''

def req(main_link, pages):
        '''
        This function works with the search link and pushes input into the above functions
        @param main_link: an url string adress
        @param pages: number of pages to work with
        '''
        s_freq_dict = {}
        pr_freq_dict = {}
        v_freq_dict = {}
        a_freq_dict = {}
        adv_freq_dict = {}
        constr_dict = {}
        log('req')
        # листаю страницы выдачи, запоминаю их номера
        for i in range(pages):
```

```
        log('I got page %s'%i)
        # делаю нужное количество попыток соединения с сайтом
        for n_try in range(MAX_RETRY):
        try:
        # если получилось, пытаюсь найти все выделенные цветом слова - они
мои будущие конструкции
        all_highlighted = search_highlighted(main_link+'&p=%s' % i, s_freq_dict,
pr_freq_dict, v_freq_dict, a_freq_dict, constr_dict, adv_freq_dict)
        break
                # эта ссылка была помечена как ошибка, ссылка рабочая.
Сообщение: не удалось загрузить внешний элемент
        except OSError:
        raise
        # если что то пошло не так, поднимаю ошибку и делаю паузу
        time.sleep(PAUSE_AFTER_FAILURE)
        log('BAD OSError happened!')
# тут я сортирую и пишу в результирующие файлы свои частотные
словарики
pr_freq_dict_sorted = OrderedDict(sorted(pr_freq_dict.items(), key = lambda t:
t[1], reverse=True))
with open('Prep_dict.csv', 'w') as csv_file:
 writer = csv.writer(csv_file, delimiter= ';')
 for key, value in pr_freq_dict_sorted.items():
        writer.writerow([key,value])
s_freq_dict_sorted = OrderedDict(sorted(s_freq_dict.items(), key = lambda t:
t[1],reverse=True))
with open('Noun_dict.csv', 'w') as csv_file:
 writer = csv.writer(csv_file, delimiter= ';')
 for key, value in s_freq_dict_sorted.items():
        writer.writerow([key,value])
v_freq_dict_sorted = OrderedDict(sorted(v_freq_dict.items(), key = lambda t: t[1],
reverse=True))
with open('Verb_dict.csv', 'w') as csv_file:
 writer = csv.writer(csv_file, delimiter= ';')
 for key, value in v_freq_dict_sorted.items():
        writer.writerow([key,value])
constr_dict_sorted = OrderedDict(sorted(constr_dict.items(), key = lambda t: t[1],
reverse=True))
with open('CONSTR_dict.csv', 'w') as csv_file:
```

```
 writer = csv.writer(csv_file, delimiter= ';')
 for key, value in constr_dict_sorted.items():
 writer.writerow([key,value])
adv_freq_dict_sorted = OrderedDict(sorted(adv_freq_dict.items(), key = lambda t: t[1], reverse=True))
with open('Adverb_dict.csv', 'w') as csv_file:
 writer = csv.writer(csv_file, delimiter= ';')
 for key, value in adv_freq_dict_sorted.items():
        writer.writerow([key,value])
a_freq_dict_sorted = OrderedDict(sorted(a_freq_dict.items(), key = lambda t: t[1], reverse=True))
with open('Adjective_dict.csv', 'w') as csv_file:
 writer = csv.writer(csv_file, delimiter= ';')
 for key, value in a_freq_dict_sorted.items():
        writer.writerow([key,value])
# это моя ссылка поиска и количество страниц выдачи
req('http://processing.ruscorpora.ru/search.xml?flags2=&spp=100&text=lexgramm&kwsz=5&mysent=&level1=0&level2=1&level3=1&level4=2&type4=&spd=10&seed=7607&type3=&out=normal&flags4=&flags1=&flags3=&mysize=&mysentsize=&simple=1&env=alpha&type2=&parent2=1&link4=on&link3=on&link2=on&gramm4=S&gramm1=V&gramm2=S&gramm3=PR&min2=&min3=&min4=&lang=ru&lex4=&lex1=&lex3=&lex2=&max2=1&max3=2&dpp=100&mycorp=&max4=3&notag=1&parent4=3&parent3=1&mode=syntax&parent1=0' ,7)
```

Приложение 2. Часть полученных конструкций и списков лемм по первому запросу.

явилась ему в момент	[['являться'	'он'	'в'	'момент']]
является лидером на рынке	[['являться'	'лидер'	'на'	'рынок']]
является Буран для космонавтики	[['являться'	'буран'	'для'	'космонавтика']]
явил себя в мере	[['являть'	'себя'	'в'	'мера']]

экстрадировать его из Австрии	[['экстрадировать'	'он'	'из'	'австрия']]
экранизированная Спилбергом в фильме	[['экранизировать'	'спилберг'	'в'	'фильм']]
эвакуировать людей от источника	[['эвакуирова'	'человек'	'от'	'источник']]
Шалит народ в уезде	[['шалить'	'народ'	'в'	'уезд']]
шагнули пылесосы за десятилетия	[['шагнуть'	'пылесос'	'за'	'десятилетие']]
чувствует человек в море	[['чувствовать'	'человек'	'в'	'море']]
чувствовали себя при этом	[['чувствовать'	'себя'	'при'	'это']]
чувствовал себя в роли	[['чувствовать'	'себя'	'в'	'роль']]
чувствую себя в лесу	[['чувствовать'	'себя'	'в'	'лес']]
чувствовала его в взвизге	[['чувствовать'	'он'	'в'	'взвизг']]
читал лекции в институте	[['читать'	'лекция'	'в'	'институт']]
чистить тело при помощи конечностей	[['чистить'	'тело'	'при помощи'	'конечность']]
чистил ботинки на рынке	[['чистить'	'ботинок'	'на'	'рынок']]
чествовать их на неделе	[['чествовать'	'они'	'на'	'неделя']]
чешет всех под гребенку	[['чесать'	'все'	'под'	'гребенка']]
поцеловал меня в губы	[['целовать'	'я'	'в'	'губа']]

Приложение 3. Часть частотного словаря глаголов для первого запроса.

быть	188
идти	102
получать	99
приводить	95
происходить	77
делать	75
находиться	73
связывать	71
брать	69
приходить	67
проходить	66
использовать	61
иметь	61
проводить	61
говорить	59
оставаться	56
оказываться	52
становиться	51
ставить	49
заявлять	46
начинаться	45

выходить	42
видеть	42
направлять	42
сообщать	41
уходить	40
принимать	39
состоять	39
пойти	39

Приложение 4. Алгоритм для объединения частотныйх словарей.

```
import csv
# для перечисления всех файлов из папки
import os
# для сортировки словаря
from collections import OrderedDict
# новый словарь, где будут все словари в объединенном виде
output = {}
# перебираю все файлы из директории
for filename in os.listdir(os.getcwd()):
 if not filename.endswith('.csv'):
        continue
 file = open(filename, 'r')
 # считываю содержимое файла как словарь, учитывая разделитель
 dic = csv.reader(file, delimiter=';')
 # перебираю ключи и значения словаря
 for string in dic:
 # ключ это слово
 key = string[0]
 # значение это частота слова
```

```
 value = int(string[1])
 # если ключа нет в словаре, добавляю ключ и его значение
 if key not in output:
        output[key] = value
 else:
        # если ключ уже есть, суммирую старое значение и новое
        output[key] = output[key] + value
# сортирую словарь и записываю в результирующий файл csv с разделителем
;
result = OrderedDict(sorted(output.items(), key = lambda t: t[1], reverse=True)) #
в данном случае объединяем словари глаголов
with open('Verb_Total.csv', 'w') as csv_file:
        writer = csv.writer(csv_file, delimiter= ';')
        for key, value in result.items():
        writer.writerow([key,value])
```

Приложение 5. Алгоритм получения общего списка конструкций.

```
import csv
# для перечисления всех файлов из папки
import os
# для сортировки словаря
from collections import OrderedDict
# новый словарь, где будут все словари в объединенном виде
output = {}
# перебираю все файлы из директории
for filename in os.listdir(os.getcwd()):
 if not filename.endswith('.csv'):
        continue
 file = open(filename, 'r')
 # считываю содержимое файла как словарь, учитывая разделитель
 dic = csv.reader(file, delimiter=';')
 # перебираю пары ключ-значение словаря
```

```
 for string in dic:
        # print(string)
        # ключ это конструкция
        key = string[0]
        # print(key,'key')
        # значение это список лемм
        value = string[1]
        # print(string[1])
        # print(type(value))
        # если ключа нет в словаре, добавляю ключ и его значение
        if key not in output:
        output[key] = value
        # print(output[key],' output[key]')
# сортирую словарь и записываю в результирующий файл csv с разделителем
;
result = OrderedDict(sorted(output.items(), key = lambda t: t[1], reverse=True))
with open('TOTAL.csv', 'w') as csv_file:
 writer = csv.writer(csv_file, delimiter= ';')
 for key, value in result.items():
        writer.writerow([key,value])
```

Приложение 6. Алгоритм получения выборки конструкций.

```
import csv
# для перечисления всех файлов из папки
import os
# для сортировки словаря
from collections import OrderedDict
# результирующий словарь
output = {}
# будущий список списков
total_list = []
# файл, куда пишется вся выдача
```

```
log_file = open('result.log', 'w')

def log(*args):
    '''
    This function writes information in file and also prints it on the screen
    @param *args: arguments; the number of arguments is not limited
    '''
    log_file.write(' '.join(str(arg) for arg in args))
    log_file.write('\n')
    log_file.flush()
    '''print(*args)'''

def lemma_generator(lists):
    '''
 This function generates lemmas
    @param lists: list of lists of lemma:[frequency,tag] items
    It chooses a lemma with a max frequency upon firsts elements of each list and yeilds it
    '''
    # превращаю списки в итераторы. итератор - генератор, осуществляющий итерацию.
    iters = [iter(x) for x in lists]
    # список с первыми элементами списков
    firsts = [next(it) for it in iters]
 while firsts:
     # нахожу лемму с максимальным значением частоты
     lemma_iter = max(firsts, key=lambda lem_tuple: (int(lem_tuple[1])))
     yield (lemma_iter[0],lemma_iter[1],lemma_iter[2])
     # номер массива, из которого я взяла этот элемент
     lemma_iter_pos = firsts.index(lemma_iter)
```

```
            try:
            # переходим к следующему элементу в этом списке
            firsts[lemma_iter_pos] = next(iters[lemma_iter_pos])
            except StopIteration:
            # если один из списков закончился, то удаляем и возвращаем удаленный элемент
            # те первый элемент такого списка и его итератор нам больше не нужны и мы их удаляем
            iters.pop(lemma_iter_pos)
            firsts.pop(lemma_iter_pos)


# в этом блоке кода я делаю список списков, который потом передам функции def lemma_generator(lists)
# перебираю все файлы из директории, где есть частотные словари
for filename in os.listdir(os.getcwd()):
 # список всех троек лемма-частота-тег для каждого файла
 dictlist = []
 # тк перебираются все файлы, включая данный, ставлю проверку на нужные файлы, те файлы csv
        if not filename.endswith('_1.csv'):
         continue
        file = open(filename, 'r')
        # считываю содержимое файла как словарь, учитывая разделитель
        dic = csv.reader(file, delimiter=',')
        for string in dic:
         # lemma это лемма слова
         lemma = string[0]
         # часота слова
         freq = string[1]
         # грамматический тег слова
```

```
            tag = string[2]
            # print(tag,'tag')
            # добавлю лемму, частоту и тег в список
            dictlist.append((lemma,freq,tag))
            # делаю список списков
           total_list.append(dictlist)
# множество пройденных лемм
already_found_lemmas = set()
# already_found_lemmas = {}
freq_test = []
# множество конструкций, в которые вошли все леммы из множества лемм
constr_set = list()
# множество лемм, которые я буду обрабатывать в онтологии
lemmas_for_ontology = set()
# результирующий файл со списком конструкций
final_list_constr = open(«fina_list_constr.txt»,  «w«)
# финальный список лемм, которые надо проверить в онтологии
final_list_of_lemmas = open(«lina_lemmas_list.txt», «w«)
# сумма частот в процессе генерации лемм
summary = 0
# общая сумма всех частот по всем частям речи
total = 516562
# читаю файл по строчкам
stop_data = stop_constr_list.readlines()
# удаляю символ \n
stop_data = [line.rstrip() for line in stop_data]
# сюда запишу словарь вида лемма - частота,
for lemmas_info in list(lemma_generator(total_list)):
 lemma = lemmas_info[0]
 freq = int(lemmas_info[1])
```

```
 tag = lemmas_info[2]
 summary += freq
        if float(summary)/ total >= 0.8:
         break
        if lemma not in already_found_lemmas:
         already_found_lemmas.add(lemma)
         freq_test.append(freq)
         already_found_lemmas[lemma] = list()
         already_found_lemmas[lemma].append(freq)
         already_found_lemmas[lemma].append(tag)
 # открываю файл с конструкциями и леммами
         constrs = open('TOTAL.csv', 'r')
         for constr in constrs:
         constr, lemmas = constr.split(«;«)
         if constr not in stop_data:
         lemmas = eval(lemmas)
         constr_ok = True
         for constr_lemma in lemmas[0]:
         if constr_lemma not in already_found_lemmas:
         constr_ok = False
         break
         if constr_ok and constr not in constr_set:
         log («i am here«)
         for constr_lemma in lemmas[0]:
         if constr_lemma not in lemmas_for_ontology:
         lemmas_for_ontology.add(constr_lemma)
         final_list_of_lemmas.write(constr_lemma + '\n' )
         constr_set.append(constr)
         final_list_constr.write(constr + '\n')
# не забудь закрыть файл
```

final_list_constr.close()

final_list_of_lemmas.close()

stop_constr_list.close()

Приложение 7. Часть конструкций из результирующей выборки

уровень жизни в стране

уровень жизни в России

решения проблем в России

получая деньги по месту

решения проблем в отношениях

отношений России с странами

развития в стране системы

Это процесс по делу

развития процессов в странах

сделаем все для решения

программа развития на годы

делать деньги в условиях

развитием ситуаций на Земле

имело отношение к борьбе

имеем дело с людьми

имеет дело с частью

тысяч школ по стране

стать проблемой для себя

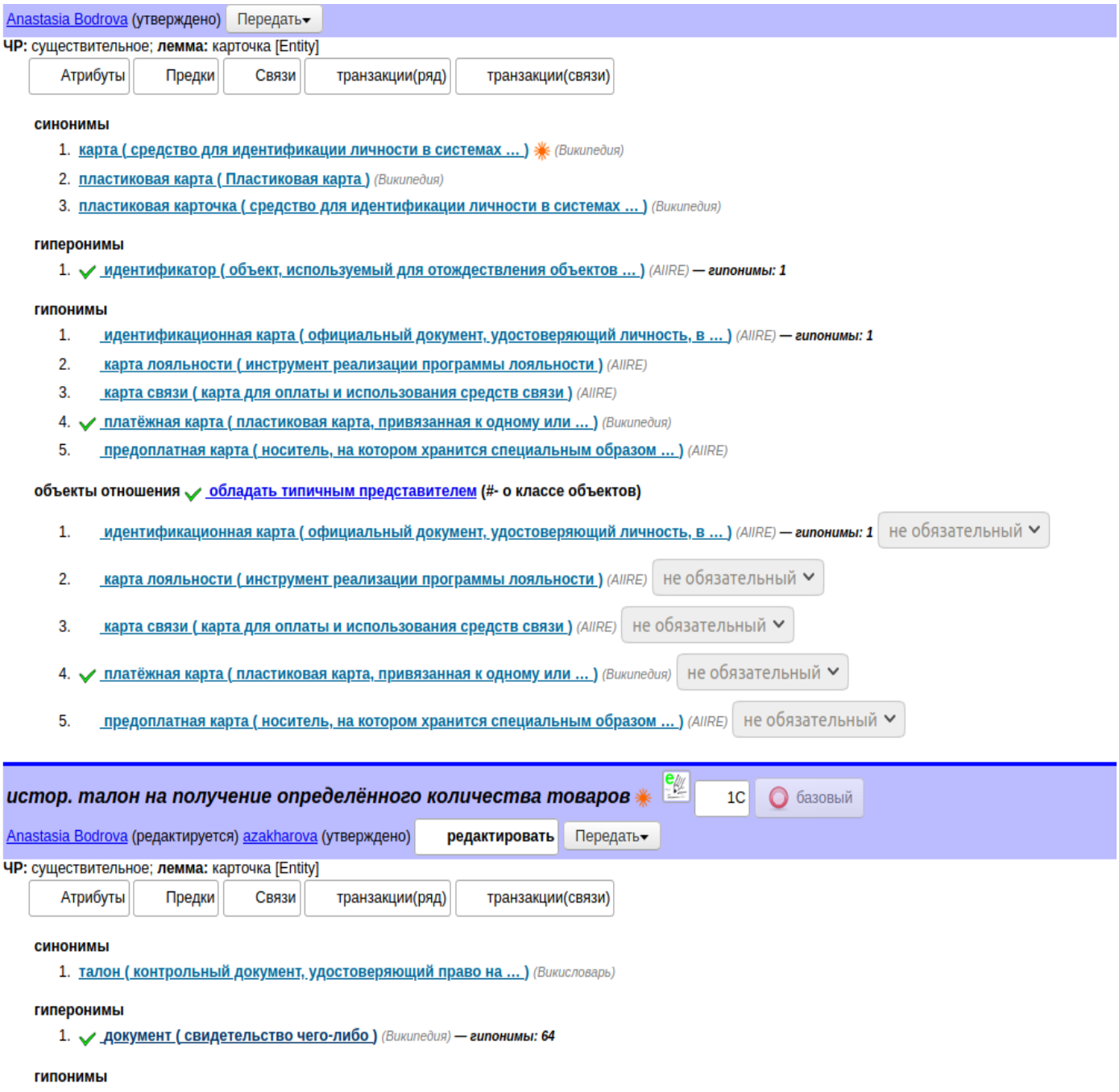
приход на пост человека

начаты переговоры с партнерами

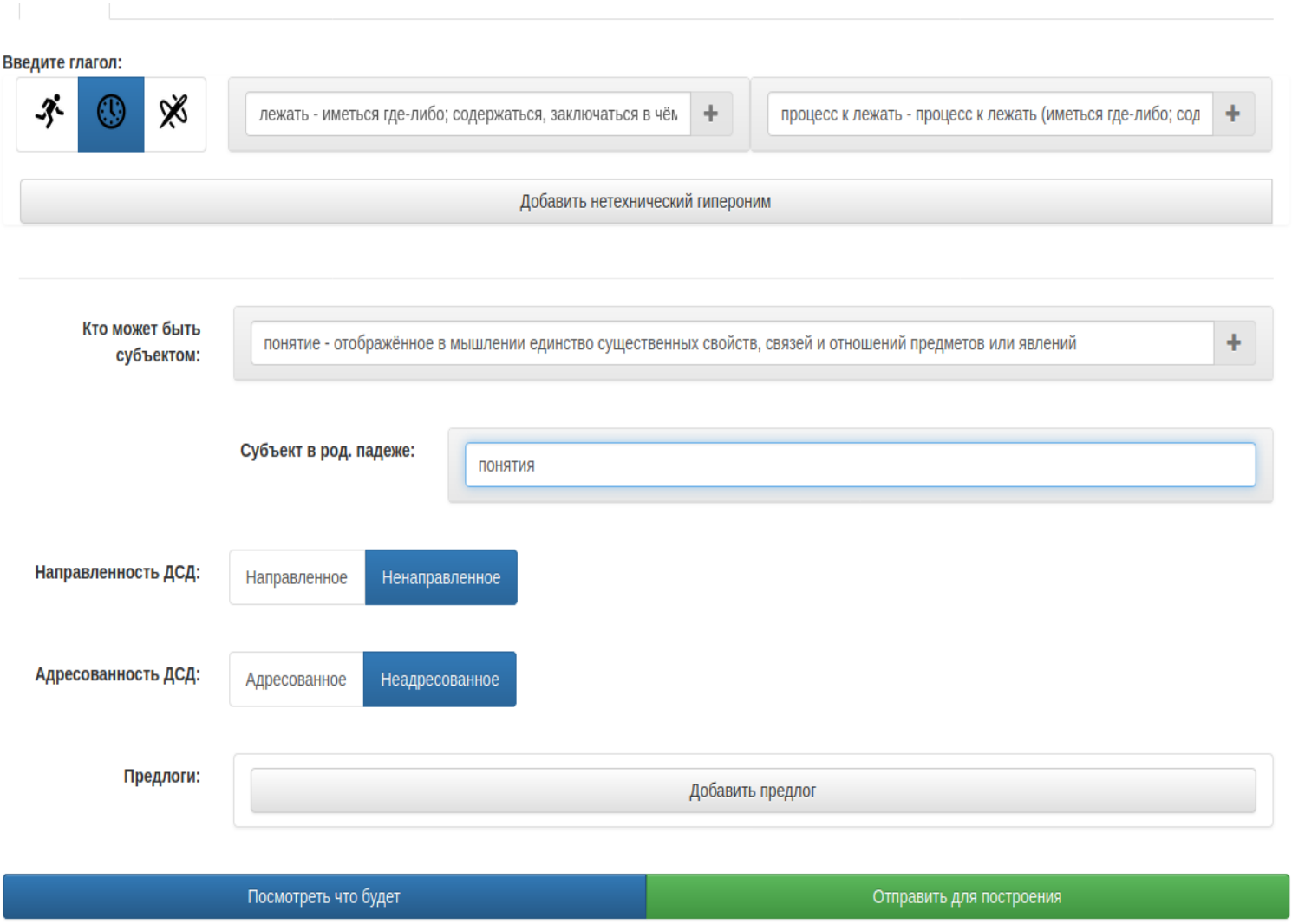
организовавшие теракт на острове

система определения по состоянию

Приложение 8. Пример работы с онтологией AIIRE.

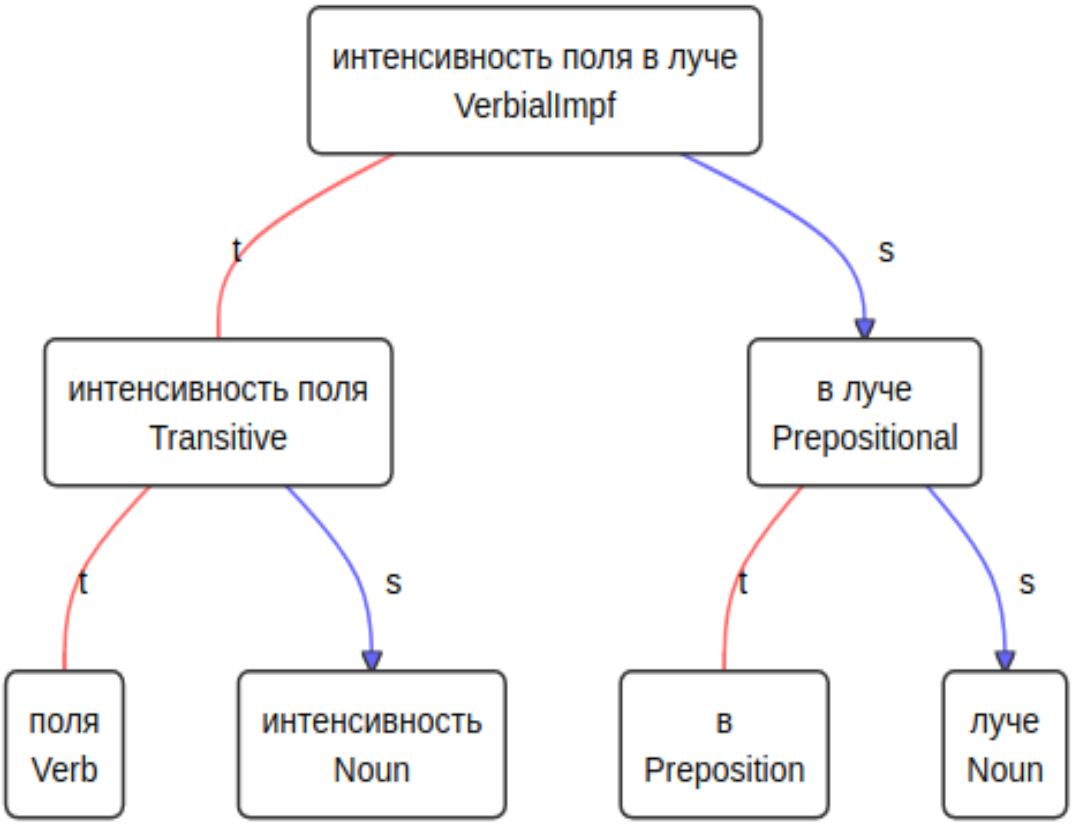


Приложение 9. Пример работы в Ontohelper.



Приложение 10. Пример синтаксического и семантического разбора в корпус-менеджере AIIRE.

Синтаксическое дерево к конструкции «интенсивность поля в луче»



Семантический граф к конструкции «интенсивность поля в луче (их 4, это первый)