Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра информационных и управляющих систем

Работа допущена к защите
Зав. кафедрой ИУС
_____П.Д. Дробинцев
«_____»_____ г.

ВЫПУСКНАЯ РАБОТА БАКАЛАВРА

Тема: **Разработка метода и программного модуля для генерации достоверных гипотез пользовательского ввода в системах понимания естественной речи**

Направление**:** 09.03.01 – Информатика и вычислительная техника

Выполнил: студент гр. 43504/3 Е.О. Грицина

Руководитель: ст. преподаватель Т. В. Коликова

Санкт-Петербург
2016 г

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой ИУС
П.Д. Дробинцев
«_____»_____г.

З А Д А Н И Е
к работе на соискание степени бакалавра

Студенту группы 43504/3 Грицина Егору Олеговичу

Тема проекта (работы) Разработка метода и программного модуля для генерации достоверных гипотез пользовательского ввода в системах понимания естественной речи

(утверждена распоряжением по ИКНТ от _____№_____)

2. Срок сдачи студентом оконченного проекта (работы)

3. Исходные данные к проекту (работе)
Языки программирования: Java
Средства разработки: IntelliJ IDEA the Java IDE
Системные библиотеки: Stanford CoreNLP tools

4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)

Обоснование необходимости генерации гипотез пользовательского ввода
Анализ существующих подходов к генерации гипотез пользовательского ввода
Определение требований к разрабатываемому методу генерации гипотез
Разработка метода составления гипотез пользовательского ввода
Программная реализация модуля составления гипотез на основе синтаксического разбора предложения
Разработка метода оценки достоверности гипотезы
Программная реализация модуля оценки достоверности гипотезы
Анализ результатов работы

5. Перечень графического материала (с точным указанием обязательных чертежей)

_____
_____
_____
_____
_____
_____
_____

6. Консультанты по проекту (с указанием относящихся к ним разделов проекта, работы)

_____
_____
_____
_____
_____

7. Дача выдачи задания _____

Руководитель Коликова Т.В.

Задание принял к исполнению _____

(дата)

_____

(подпись студента)

ПРИМЕЧАНИЯ: 1.Это задание прилагается к законченному проекту (работе) и вместе с проектом представляется в ГАК.

2. Кроме задания, студент должен получить от руководителя календарный график работы над проектом (работой) на весь период проектирования (с указанием сроков выполнения и трудоемкости отдельных этапов).

Реферат

Работа содержит 104 страницы, 5 иллюстраций, 2 таблицы и 3 приложения. При написании работы использовано 7 источников.

Обработка естественной речи, синтаксический анализ, синтаксические деревья, грамматические шаблоны, Stanford CoreNL, The Stanford Parser.

Данная работа посвящена такой области информационных технологий, как обработка и понимание естественной речи. Целью работы является разработка алгоритма, позволяющего решить проблему выделения смысла из предложений, в которых используются лишние распространяющие слова. Разработанный модуль позволяет сократить и упростить предложения, которые не подходят под существующие грамматические шаблоны.

Разработанный алгоритм успешно справляется со своей задачей и может быть внедрен в существующие системы выделения смысла из предложений.

Содержание

Список иллюстраций

Список таблиц

Введение

В настоящее время бурное развитие компьютерной техники и информационных технологий привело к тому, что компьютер, мобильные устройства и информационные технологии присутствуют во всех сферах жизни современного человека. Компьютеры и мобильные устройства стали неотъемлемой частью современного мира. Развитие компьютерной техники с резко ускорилось и происходит невероятно быстрыми темпами и на сегодняшний день. Столь же стремительно развивается и процесс миниатюризации компьютеров. Первые электронно-вычислительные машины были огромными устройствами, весившими тонны, занимавшими целые комнаты и требовавшими большого количества обслуживающего персонала для успешного функционирования. В контрасте с этим, современные компьютеры — гораздо более мощные и компактные и гораздо менее дорогие — применяются повсеместно.

Такое стремительное и быстрое развитие компьютеров и уменьшение их размеров приводит к тому, что привычные для нас способы взаимодействия

человека с компьютером становятся все менее удобными. И перед производителями современных устройств и компьютерных систем возникла новая задача – создание нового способа человеко-компьютерного взаимодействия. Взаимодействие между пользователем и компьютером традиционно происходит с помощью различного программного и аппаратного обеспечения, например, с помощью образов или объектов на экране, или с помощью данных, полученных от пользователя посредством аппаратных устройств ввода (клавиатура, мышь).

Человеко-компьютерное взаимодействие – это отдельное научное направление, которое существует и развивается в целях совершенствования методов взаимодействия человека (пользователя) и компьютера. Основной задачей этого научного направления является улучшение этого самого взаимодействия, снижение барьера между человеческой моделью того, чего хотят достичь пользователи, и пониманием компьютера поставленных перед ним задач. Взаимодействие с компьютером с помощью экрана, мышки и клавиатуры

на протяжении долгого времени отлично справлялось с этой задачей.

Вернемся к проблеме уменьшения размеров устройств. Уже сейчас на рынке пользуются большой популярностью «умные» часы, фитнесс-браслеты, миниатюрные музыкальные плееры, устройства «умного дома». Все эти устройства могут поместиться на ладони человека. И управление ими с помощью клавиатуры или мышки (пусть даже сенсорного экрана) трудно назвать удобным. По этой причине, в области человеко-компьютерного взаимодействия появился новый интерфейс – управление устройствами с помощью голосовых команд.

Идея управления компьютером с помощью голосовых команд за последнее время развилась в совершенно новую сферу информационных технологий, о которой раньше можно было лишь прочитать в книжках в жанре научной фантастики. Название этой новой области информационных технологий – обработка естественного языка (*Natural Language Processing, NLP*) или понимание естественного языка (*Natural Language Understanding, NLU*). То есть

общение с компьютером и управление им с помощью естественной речи так, будто пользователь общается с таким же человеком.

Актуальность темы

Как уже было сказано выше, понимание естественной человеческой речи – это совершенной новый способ взаимодействия человека и компьютера, который призван облегчить обмен информацией между пользователем и устройством и сделать его максимально удобным и привычным для человека.

Обработка естественного языка - совместное направление искусственного интеллекта и математической лингвистики. Оно изучает проблемы компьютерного анализа и синтеза естественных языков. Применительно к искусственному интеллекту анализ означает понимание языка, а синтез — генерацию грамотного текста. Решение этих проблем будет означать создание более удобной формы взаимодействия компьютера и человека. Подобными исследованиями занимается огромное количество компаний и небезуспешно. Достаточно вспомнить такие популярные голосовые помощники на мобильных

устройствах, как Siri от корпорации Aplle, Cortana от Microsoft, Google Now от Google, Alexa от Amazon, и бесконечное количество чат-роботов, например в сервисе Telegram.

Алгоритм работы всех этих систем можно представить следующим образом:

1. Распознавание речи. В первую очередь необходимо распознать, что именно сказал пользователь, получить текстовую фразу из голосовой команды.
2. Анализ текста. На этом этапе из сказанной пользователем фразы необходимо извлечь информацию, проанализировать ее, проанализировать характер высказывания, тональность текста, извлечь «смысл» сказанного, понять, чего именно хочет пользователь.
3. Генерация текста. На основе анализа пользовательского текста необходимо сформировать какой-то ответ системы на запрос. Так как речь идет о взаимодействии с пользователем путем естественной речи, то и ответ компьютера тоже должен быть сформирован в виде текста на естественном языке.

4. Синтез речи. Компьютеру или мобильному устройству необходимо произнести подготовленный ответ.

Очевидно, что основной проблемой при разработке систем понимания естественной речи является понимание смысла сказанной человеком фразы и генерация грамотного, логичного ответа для пользователя. И решение этих проблем крайне актуально и необходимо, если мы и дальше хотим создавать новые, более компактные, удобные и умные устройства.

Поэтому, в своей работе я предлагаю решение одной из проблем, с которой сталкиваются разработчики систем понимания естественной речи, на этапе понимания сказанного пользователем текста и извлечения из него информации для дальнейшего анализа.

Формулировка проблемы

В современных информационных технологиях роль такой процедуры, как извлечение информации, всё больше возрастает. А в такой научной области как

понимание естественного языка процесс извлечения информации из сказанного пользователем текста играет ключевую роль. Примером извлечения информации может быть поиск некоторых информационных конструкций — формально это записывается так: *НанеслиВизит(Компания-Кто, Компания-Кому, ДатаВизита),* — из новостных лент, таких как: «Вчера, 1 апреля 2007 года, представители корпорации Пепелац Интернэшнл посетили офис компании Гравицап Продакшнз». Главная цель такого преобразования — возможность анализа изначально «хаотичной» информации с помощью методов обработки данных. Решением такой задачи занимается компьютерная лингвистика - научное направление в области моделирования интеллектуальных процессов при создании систем искусственного интеллекта, которое ставит своей целью использование математических моделей для описания естественных языков.

Процесс извлечения информации (смысла) из сказанного пользователем текста, по сути, сводится к извлечению из него следующих смысловых сущностей:

1. Извлечение намерения пользователя (Intents). Намерения представляют собой отображение того, что сказал пользователь, и какие меры должны быть приняты вашим программным обеспечением.
2. Извлечение действия (Actions). Действия соответствуют шагам, которые ваше приложение будет предпринимать, когда пользователь выразит определенные намерения. Действие может иметь параметры для указания некоторой информации для приложения.
3. Сохранение и извлечения контекста (Contexts). Контекст представляет собой историю диалога с пользователем, которая позволяет точнее определить смысл текущей фразы и дифференцировать различные намерения и действия пользователя в зависимости от того, что было сказано ранее.

Извлечение из текста (произнесенной фразы) причисленных выше смысловых сущностей – задача всех систем, которые занимаются пониманием естественной человеческой речи. И этот процесс основывается практически на одних и тех же

алгоритмах обработки данных. Рассмотрим принцип работы большинства таких систем и определим проблемы, с которыми они сталкиваются.

Процесс определения намерений пользователя и требуемых от вашего приложения действий основан на описании некоторых грамматических шаблонах и словарей для синтаксиса естественного языка. Рассмотрим этот алгоритм на примере следующей фразы:

Can you describe witness in case 12?

Для того чтобы система могла определить смысл сказанной фразы, необходимо описать некоторые шаблоны, которые будут относиться к одному конкретному намерению пользователя, назовем это намерение (Intent) – *DescribeWitness*. Придумаем шаблон для конкретно этого намерения:

@canyou @describe @witness @in @incident @number

Где символ «@» обозначает словарь, например, @canyou={can you tell, please tell me, give me, please, @null…}, @incident={case, incident, occasion…},

@describe={describe, give, show…}, @witness={witness, eyewitness, bystander} и так далее.

Таким образом, описав некий грамматический шаблон и заполнив определенные словари, из которых состоит этот шаблон, мы научили систему определять намерение *DescribeWitness* пользователя для довольно широкого диапазона фраз. Под данный грамматический шаблон, очевидно, можно отнести следующие высказывания: *please describe witness in incident 12, show me witness in case 12, show eyewitness in incident 12* и так далее.

Получается, что системы, которые разрабатываются для понимания естественного языка, в своей работе опираются на описание грамматики языка в виде **словарей** и **шаблонов**, состоящих из этих словарей. И весь алгоритм понимания сказанной фразы заключается в том, чтобы определить – под какой именно грамматический шаблон попадает эта фраза, а уже после того, как будет определен шаблон, можно решить, какое намерение и действие несет в себе произнесенная фраза.

Очевидно, что системы, работающие по такому алгоритму, нуждаются в длительном и тщательном обучении. И смысловая составляющая сказанной пользователем фразы может быть не определена по двум причинам: в системе не описан подходящий шаблон или словари, использующиеся в шаблонах, недостаточно распространены и слова, используемые в предложении, не могут быть отнесены к какому-либо словарю. То есть, для повышения точности выделения смысла из текста, разработчикам системы постоянно необходимо создавать новые шаблоны и расширять существующие словари. Это, пожалуй, единственный и самый логичный способ решения данной проблемы – просто расширять существующий шаблон путем обновления словарей.

Вторая проблема, по которой не представляется возможным определить смысл сказанной фразы – это случай, когда данная фраза распространена дополнительными словами, появление которых в предложении невозможно было предсказать на этапе написания грамматического шаблона. Такая фраза в принципе не может соответствовать существующим на

данный момент шаблонам. Данную проблему уже не решить путем расширения словарей, в этом случае необходимо создавать уже новый шаблон и даже новые словари. Например, распространим выше описанный пример: *Can you describe **second** witness in case 12 **for me**?*

Такая фраза уже не подходит под созданный нами шаблон, и для определения ее смысла необходимо создать новый. Очевидно, что для определения одного и того же намерения (Intent) пользователя может потребоваться создать бесконечное число таких шаблонов, и даже в этом случаи мы не сможем предусмотреть все возможные варианты формулировок запроса пользователя. Для решения этой проблемы необходимо изменять сам запрос пользователя – исключать из него «мешающие» слова, производить сокращение фразы, использовать суммаризацию, регулярные выражения и так далее. В общем смысле – необходимо сформировать **гипотезы** или варианты того, что хотел сказать пользователь на основе исходной фразы, которые бы позволили отнести запрос к уже существующим грамматическим шаблонам.

Цели и задачи

Целью работы является разработка алгоритма в виде самостоятельного программного модуля, который бы производил формирование гипотез пользовательского ввода на основе сказанного пользователем предложения и оценку достоверности этих гипотез. Данный алгоритм позволил бы системам понимания естественной речи определить семантическое значение сказанной пользователем фразы без написания дополнительных грамматических шаблонов или расширения существующих словарей.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- Проанализировать существующие способы решения проблемы определения смысловой составляющей из предложений, распространенных излишними описательными словами.
- Определить требования к разрабатываемому алгоритму.

- Разработать алгоритм формирования гипотез таким образом, чтобы сформированные гипотезы сохраняли семантическое значение и синтаксическую целостность исходного предложения.
- Разработать метод расчета достоверности сформированных гипотез
- Реализовать программный модуль разработанного алгоритма.
- Провести тестирование разработанного алгоритма и оценить результаты его работы.

Краткое содержание работы

Работа содержит 5 основных частей.

В первой части проводится анализ предметной области и обзор существующих способов решения сформулированной проблемы. По результатам этого анализа формулируются требования к разрабатываемому алгоритму и его программной реализации.

Во второй части работы описываются теоретические основы реализации модуля формирования гипотез пользовательского ввода и приводится общая архитектура проекта.

В третье части работы приводятся подробности программной реализации разработанного модуля.

Четвертая часть работы посвящена описанию способа оценки достоверности сформированных гипотез и особенностям программной реализации этого способа.

Пятая и заключительная часть работы посвящена тестированию реализации и анализу результатов работы алгоритма.

1. Анализ предметной области

Целью моей работы является создание программного модуля, который бы производил генерацию достоверных гипотез пользовательского ввода. То есть, разработка модуля, который бы создавал из исходной фразы различные ее варианты до того, как эта фраза будет проанализирована на совпадение с различными шаблонами для извлечения смысла. Такой подход позволяет проверять на совпадение с грамматическими шаблонами не одну единственную оригинальную фразу пользователя, а сразу целый набор фраз, сформированных из исходной для того, чтобы исключить невозможность определения намерений пользователя по причине невозможности отнести фразу к какому-либо шаблону из-за распространяющих слов. Есть несколько способов решения этой проблемы. Рассмотрим некоторые из них

1.1. Игнорирование слов в шаблоне

Самый простой способ решить подобную проблему – это использование свободной позиции слов в предложении. Например, символ «*», как и в грамматике регулярных выражений, подразумевает собой наличие на его месте в исходной фразе любого количества других слов.

Понимание естественной речи во всех системах грамматического моделирования происходит путем описания шаблонов и словарей для возможных фраз пользователей. Такие платформы позволяют использовать свободную позицию слов при написании этих шаблонов, подразумевая, что в грамматическом шаблоне в определенном месте могут находиться любые другие слова.

Конечно, этот метод позволяет справиться с наличием лишний дополняющих слов в предложении и решить проблему бесконечного создания новых шаблонов. Но использование такого способа влечет за собой вполне логичные последствия – написанный таким образом шаблон начинает ошибочно применяться к совершенно неподходящим фразам, что приводит к

большому количеству ложных срабатываний для такого шаблона.

Чтобы продемонстрировать явный недостаток такого метода напишем шаблон для модели, которая позволяет распознавать намерение пользователя получить информацию о потерпевшем, например следующей фразой:

Describe witness.

Для того, чтобы распознать намерение в распространённых фразах, таких как : *describe second witness in case, describe female witness in last case, describe second or first witness* – необходимо написать следующий шаблон:

*« * @describe * @witness * »*

Очевидно, что такая модель будет иметь огромное количество ложных срабатываний на тех фразах, который попадают под данный шаблон, но совершенно не несут в себе намерения получить информацию о свидетеле. Примером таких фраз будут:

Describe vehicle of the witness in this case. Describe incident with witness. Describe route to the witness.

Каждая из этих фраз будет распознана как намерение получить информацию о свидетеле, что является неверным определением смысла сказанного и демонстрирует очевидный недостаток этого метода.

Похожий способ исключения «мешающих» слов из предложения используется в AIML (*Artificial Intelligence Markup Language* - язык разметки для искусственного интеллекта). Этот язык разметки был разработан еще в 1966 году с появлением первого виртуального собеседника – программы ALICE. Создатели языка предложили описывать логику общения чат-робота наборами образец-шаблон. И если сказанная пользователем фраза совпадает с описанным образцом, то робот выдает в качестве результата одну из фраз, записанную как шаблон для этого образца. Такой просто способ «мышления» робота логично столкнулся с теми же проблемами – оказалось, что невозможно предусмотреть все возможные варианты обращений

пользователя. И было предложено использовать так называемые сокращения или **Reductions**.

Смысл сокращений в AIML заключается в том, что при совпадении сказанной пользователем фразы с некоторым шаблоном, который обычно написан с использованием того же символа «*», все слова подходящие под этот шаблон удаляются из фразы. И получившаяся таким образом фраза отправляется на вход робота, где происходит очередная попытка подобрать шаблон для нее. Продемонстрируем алгоритм сокращений на примере.

Скажем, у нашего робота есть известный шаблон «ПРИВЕТ» на который он должен ответить «И тебе привет», но пользователь говорит роботу: «Ну, привет, робот!». Очевидно, что сказанная фраза не подходит под существующий шаблон приветствия, поэтому необходимо предусмотреть сокращения для такого случая. Определим эти сокращения:

- «НУ*» - <sr/>
- «*РОБОТ*» - <sr/>

Определённые таким образом шаблоны нужно читать следующим образом: если фраза начинается на слово «НУ» или если в ней содержится слово «РОБОТ» то необходимо исключить из фразы эти слова и обработать получившуюся фразу еще раз. Именно таким образом в языке разметки для искусственного интеллекта решена проблема «лишних» слов.

Но подобный подход совсем не избавляет разработчика системы понимания естественной речи от необходимости создания бесконечного числа шаблонов. Данный способ дает возможность создать один шаблон для реакции на желаемую фразу, но обязует создать бесконечное число шаблонов для сокращения.

1.2. Методы суммаризации

Следующий способ создания гипотез пользовательского ввода – это различные методы **суммаризации текстов**. Суммаризация текста представляет собой автоматическое выделение ключевой информации из текста и создание краткого изложения для него. Идея суммаризации является довольно перспективной, учитывая повсеместное

распространение мобильных устройств и постоянное увеличение потока информации в современном мире.

Есть два основных подхода к разработке методов суммаризации: обобщение и извлечение. Обобщающие алгоритмы анализируют структуру текста, чтобы «понять», о чем он, а затем создают новый текст с основным содержанием. В общем, обобщение работает так, как делал бы живой человек. И хотя понятно, что за таким подходом будущее, сейчас подобные методы еще развиты слабо. Поэтому чаще применяются извлекающие алгоритмы, которые анализируют текст статистически, а потом выбирают из него наиболее важные куски.

Однако любой алгоритм суммаризации будет эффективен только в том случае, если его применяют к объемному тексту, статье, странице. Потому что эти алгоритмы основаны на анализе связей между несколькими предложениями текста, выделению среди них ключевых и наиболее повторяющихся слов. Например, алгоритм суммаризации *TextRank*, который основан на преобразовании текста в граф, начинает

корректно работать, только если тест содержит хотя бы 30 предложений.

Очевидно, что в системах, требующих понимания естественной речи или реализующих голосовое управление, алгоритмы суммаризации не представляется возможным применить на этапе извлечения информации из сказанной фразы. Так как в рассматриваемых системах длина сказанного текста обычно ограничивается десятком слов.

Так как ни один из рассмотренных выше вариантов не подходит для решения проблемы необходимости создания бесконечного числа шаблонов, возникает необходимость разработки иного решения этой задачи.

Алгоритм формирования гипотез пользовательского ввода, разработанный в рамках этой работы, основывается на нескольких ключевых принципах:

- Формирование гипотез пользовательского ввода основано на анализе синтаксического отношения между словами в предложении.

- Формирование гипотез ввода происходит путем исключения из предложения распространяющих слов на основе некоторых правил, позволяющих сохранить ключевые участки фразы без потери общего смысла сказанного.
- Любая система извлечения смысла из предложения принимает на вход набор «вариантов» произнесенного пользователем текста в виде пар «гипотеза - достоверность». Набор этих пар формируется любым модулем распознавания человеческой речи. Принцип действия моего алгоритма основывается на расширении этого списка гипотез путем их искусственного создания, и дальнейшей передачи нового списка гипотез на модуль извлечения смысла.
- Чаще всего пользователю достаточно получить от системы ответ на чуть более общий вопрос, чем несколько раз переформулировать свой запрос и каждый раз получать сообщение о невозможности сгенерировать точный ответ.

Таким образом, в рамках данной работы будет решена задача разработки метода и программного

модуля для генерации достоверных гипотез пользовательского ввода в системах понимания естественной речи.

1.3. Определение требований к разрабатываемому методу

Как уже было сказано ранее, все системы, занимающиеся извлечением информации из введенного пользователем текста, работают на основе схожих алгоритмов. И точность определения смысловой составляющей введенного текста в таких алгоритмах напрямую зависит от того, какое количество грамматических шаблонов и словарей создали разработчики при разработке подобной системы. В конечном итоге для максимально правильного определения смысла сказанной пользователем фразы разработчикам системы понимания естественной речи необходимо создать практически бесконечное число таких шаблонов и словарей, потому что невозможно предугадать и описать все варианты формулировок для того или иного намерения пользователя.

Существующие способы решения данной проблемы обладают очевидным недостатком –

применение их приводит к большому числу ложных срабатываний шаблонов на фразы, не несущие в себе искомого намерения. Такое поведение подобных методов вызвано тем, что эти методы основаны на простом удалении слов из фразы, без какого-либо смыслового или синтаксического анализа предложения. В данной работе предложен алгоритм, который лишен этих недостатков.

Определим набор требований, предъявляемых к разрабатываемому алгоритму:

1. Формирование гипотез пользовательского ввода должно происходить на основе исходной фразы, введенной пользователем, и до обработки ее системой извлечения информации и определения ее семантического значения.
2. Формирование новых гипотез ввода должно происходить путем вычеркивания из исходной фразы некоторого количества слов, как минимум по одному слову.
3. Каждое изменение исходной фразы пользователя должно порождать новую гипотезу, причем

достоверность этой гипотезы должна быть ниже, чем достоверность оригинальной фразы.

4. Достоверность сформированной гипотезы должна рассчитываться в зависимости от количества удаленных из исходной фразы слов и их синтаксической роли в предложении.
5. Исходная фраза, введенная пользователем, должна сохраняться в списке гипотез, причем иметь максимальную достоверность, по сравнению с искусственно сформированными гипотезами.
6. Процесс формирования новой гипотезы ввода на основе исходного предложения должен сохранять семантическое значение этого предложения.
7. Искусственно сформированные гипотезы ввода должны обладать синтаксической корректностью, при условии, что исходная фраза была изначально синтаксически корректно построена.

Таким образом, принцип действия разрабатываемого в данной работе модуля формирования гипотез пользовательского ввода можно продемонстрировать следующей схемой:

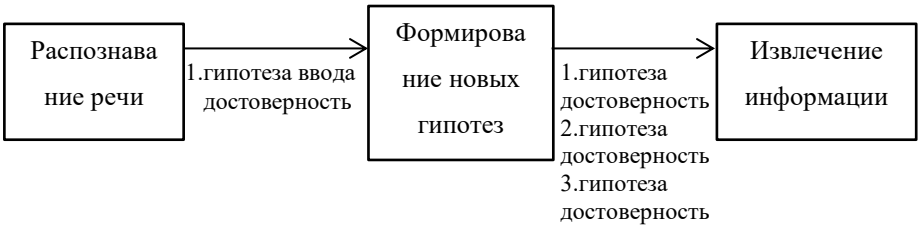

Рис. 1 Общая схема работы алгоритма

На вход модуля генерации гипотез пользовательского ввода подается информация с системы распознавания человеческой речи в виде набора гипотеза-достоверность. Но основе этого набора гипотез, разрабатываемый мной модуль должен сформировать новые гипотезы ввода, сохраняя семантической значение и смысловую корректность произнесенной пользователем фразы. Для каждой искусственно сформированной гипотезы модуль должен оценить достоверность этой гипотезы. Показатель достоверности, как и в случае с распознаванием человеческой речи, должен отражать то, насколько гипотеза соответствует произнесённой пользователем фразе. На выходе модуля должен получаться новый набор гипотез пользовательского ввода, который в

дальнейшем будет обрабатываться системой извлечения информации. При этом при работе модуля, исходная гипотеза ввода должна сохраниться в наборе и обладать максимальной достоверностью, по сравнению с искусственно сформированными гипотезами.

2. Разработка метода генерации гипотез пользовательского ввода

Разрабатываемый алгоритм формирования гипотез пользовательского ввода по результатам своей работы должен решать две задачи: искусственное создание гипотез пользовательского ввода и оценка достоверности этих гипотез. Прежде всего, необходимо разработать общий принцип действия и архитектуру алгоритма формирования гипотез.

Исходя из сформулированных требований к модулю генерации гипотез и целей всего проекта, формирование новых гипотез ввода должно происходить на основе произнесенной пользователем фразы и с сохранением ее семантического значения и синтаксической корректности. Для того чтобы добиться подобного результата – необходимо производить «вычеркивание» распространяющих и дополняющих слов в предложении на основе некоторых синтаксических правил, то есть, для корректного формирования новых гипотез, необходимо провести анализ синтаксического значения и роли каждого слова

в предложении, а уже затем принимать решение о вычеркивании этого слова из исходной фразы.

Таким образом, для того, чтобы разрабатываемый алгоритм не обладал недостатками рассмотренных в первой главе способов достижения поставленной задачи, этот алгоритм должен основываться на анализе синтаксического отношения между словами в исходном предложении, и принимать решение на вычеркивание определенного слова на основе этого анализа.

2.1. Синтаксический анализ предложений

Подобный анализ можно производить различными методами, но в данной работе за основу взят метод синтаксического анализа предложения. Синтаксический анализ предложения – процедура, которая знакома каждому человеку еще со школы и представляет собой разбор синтаксических единиц: словосочетаний и предложений. И если в естественном своем смысле процесс синтаксического разбора предложения сводится к простому задаванию наводящих вопросов, таких как – кто сделал, что сделал, как сделал и так далее, то реализация подобного

алгоритма на компьютере является достаточно трудной задачей из области лингвистики и понимания естественного языка.

Синтаксический анализ (или парсинг) в лингвистике и информатике — это процесс сопоставления линейной последовательности лексем (слов, токенов) естественного или формального языка с его формальной грамматикой. Результатом такого анализа обычно является дерево синтаксического разбора (синтаксическое дерево) предложения. Именно процесс анализа подобных синтаксических деревьев предложения я хотел бы положить в основу своего алгоритма.

Как правило, результатом синтаксического анализа является синтаксическое строение предложения, представленное либо в виде дерева зависимостей, либо в виде дерева составляющих, либо в виде некоторого сочетания первого и второго способов представления. Для проведения такого анализа существуют специализированные программы, называемые синтаксическими анализаторами или синтаксическими парсерами. Так как реализация такой

программы является довольно трудной научной задачей, было решено воспользоваться уже существующим решением в рамках данной работы.

2.2.Обзор библиотеки Stanford CoreNLP

В качестве инструмента, позволяющего произвести синтаксический разбор предложения и построение его синтаксического дерева, было решено использовать библиотеку Stanford CoreNLP, которая предоставляет собой набор инструментов для обработки текста, основанный на работах Stanford NLP Group. Стэнфордская группа обработки естественного языка (англ. *The Stanford Natural Language Processing Group*) — коллектив исследователей, разработчиков и студентов, работающих над созданием алгоритмов, позволяющих обрабатывать и понимать естественные языки. Работы коллектива ведутся как в фундаментальных областях компьютерной лингвистики, так и в её прикладных аспектах: понимание предложений, машинный перевод, вероятностный парсинг и лингвистическая разметка, информационный поиск, снятие смысловой неоднозначности, автоматическое общение.

Сама по себе библиотека предоставляет массу возможностей по обработке текста на естественном языке, с ее помощью можно производить такие действия как: графематический анализ, морфологический анализ, синтаксический анализ, извлечение именованных сущностей из текста и многое другое. При разработке алгоритма формирования гипотез пользовательского ввода в качестве инструмента для синтаксического анализа предложений используется синтаксический анализатор для предложений, который входит в эту библиотеку и называется **The Stanford Parser.**

Данный синтаксический анализатор работает по принципу разделения предложения на «составляющие», каждая из которых далее разбивается на свои составляющие – и так до тех пор, пока алгоритм не дойдет до анализа слов. Каждой составляющей присваивается своя синтаксическая роль в предложении (подлежащее, сказуемое, дополнение и так далее), а словам – часть речи. Идею составления подобного дерева хорошо иллюстрирует следующий рисунок:

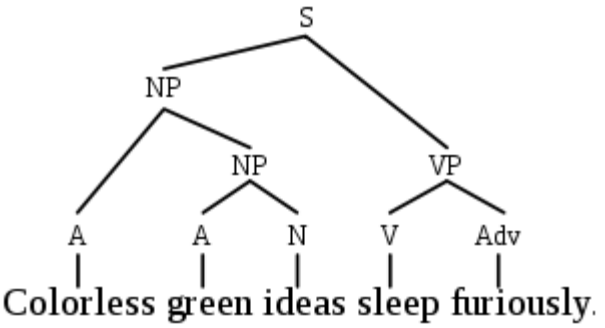

Рис. 2 Пример синтаксического дерева

Для предложения, которые мы использовали ранее, синтаксический анализатор Stanford Parse строит следующее дерево:

```
(ROOT
  (S
    (VP (VB Describe)
      (NP (NN witness))
      (PP (IN in)
        (NP (NN case) (CD 12))))))
```

На основе анализа подобных деревьев и будет работать алгоритм генерации гипотез пользовательского ввода, работа которого будет заключаться в применении некоторых правил для вычеркивания слов из синтаксического дерева и восстановления предложения по вновь сформированному дереву. Представим порядок работы алгоритма в виде схемы:

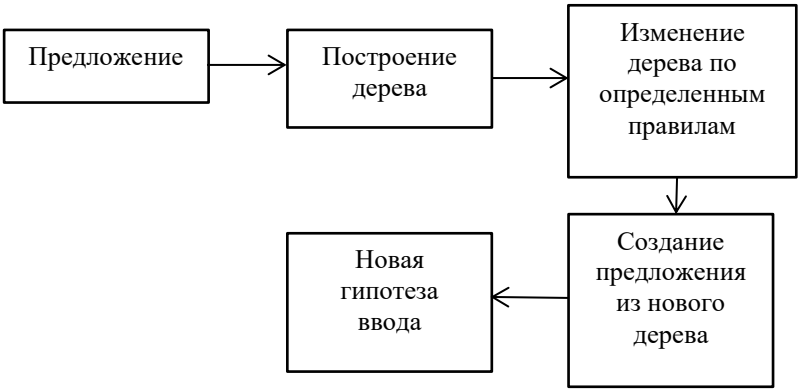

Рис. 3 Порядок формирования гипотезы из предложения

2.3.Разработка правил семантического сокращения

Правила для вычеркивания слов из предложения на этапе анализа синтаксического дерева должны быть легко изменяемыми и обособленными друг от друга. Такое свойство алгоритма позволит с легкостью расширять его возможности и подстраивать его под специфику той области, где он будет использоваться. То есть, для того чтобы «научить» алгоритм вычеркивать из предложения определенные конструкции, которые будут мешать правильному извлечению семантического значения из предложения, достаточно будет просто написать новое правило в алгоритме, и оно будет сразу же применено. На начальном этапе разработки

алгоритма и основных правил семантического сокращения предложения был взят набор фраз пользователей из реально существующей системы выделения информации из предложения, причем были взяты фразы, в которых были заранее известны лишние дополняющие слова, подлежащие вычеркиванию.

Такие фразы приведены в таблице ниже, в первом столбике таблицы приведена исходная фраза, во втором – ее синтаксическое дерево, полученное средствами синтаксического анализатора, в третьей колонке – гипотезы, которые желательно было бы получить из исходной фразы для верного определения ее семантического значения.

Таблица 1. Предложения для формирования правил сокращения и их деревья

I need to get to witness 3 in case 8776	(S (NP (PRP I)) (VP (VBP need) (S (VP (TO to) (VP (VB get) (PP (TO to) (NP (NP (NN witness) (CD 3)) (PP (IN in) (NP (NN case) (CD 8776)))))))))))	I need to get to witness in case 8776 I need to get to witness in case I need to get to witness get to witness in case 8776

Find me the quickest route to witness No 2, Jake Verdinier	(S (VP (VB Find) (S (NP (PRP me)) (NP (NP (DT the) (JJS quickest) (NN route)) (PP (TO to) (NP (NP (NP (NN witness)) (NP (DT No) (CD 2))) (NP (NNP Jake) (NNP Verdinier)))))))))	Find me the quickest route to witness No 2 Find me the quickest route to witness Jake Verdinier Find me the quickest route to witness Find me the route to witness Jake Verdinier Find me the route to Jake Verdinier Find me the route to witness
show the route to the main eyewitness for incident 7865	(ROOT (S (VP (VB show) (NP (DT the) (NN route)) (PP (TO to) (NP (NP (DT the) (JJ main) (NN eyewitness)) (PP (IN for) (NP (NN incident) (CD 7865))))))))	show the route to the eyewitness for incident 7865 show the route to the eyewitness for incident show the route to the eyewitness
describe the two male suspects in incident 769795	(ROOT (SINV (VP (VBP describe) (NP (NP (DT the) (CD two) (JJ male) (NNS suspects)) (PP (IN in) (NP (NN incident))))) (NP (CD 769795))))	describe the male suspects in incident 769795 describe the two suspects in incident 769795 describe the suspects in incident 769795 describe the suspects in incident

1990s green jeep cherokee	(ROOT (NP (NP (NNS 1990s)) (NP (JJ green) (NN jeep) (NN cherokee))))	1990 green jeep Cherokee 1990 jeep Cherokee jeep Cherokee
Tell me about robberies in 2 miles last month	(S (VP (VB Tell) (NP (PRP me)) (PP (IN about) (NP (NP (NNS robberies)) (PP (IN in) (NP (CD 2) (NNS miles))) (NP-TMP (JJ last) (NN month)))))))	Tell me about robberies last month Tell me about robberies in 2 miles Tell me about robberies
show me the quickest way to the art museum	(ROOT (S (VP (VB show) (S (NP (PRP me)) (NP (NP (DT the) (JJS quickest) (NN way)) (PP (TO to) (NP (DT the) (NN art) (NN museum))))))))	show me the way to the art museum show me the way to the museum
what's the best way to get to witness number one from here?	(ROOT (SBARQ (WHNP (WP what)) (SQ (VBZ 's) (NP (NP (DT the) (JJS best) (NN way)) (S (VP (TO to) (VP (VB get) (PP (TO to) (NP (NN witness) (NN number) (CD one))) (PP (IN from) (NP (RB here)))))))))	what's the way to get to witness number one from here what's the way to get to witness one from here what's the way to get to witness from here what's the way to get to witness

В таблице приведены лишь наиболее характерные предложения, всего же было использовано для анализа и составления правил семантического сокращения порядка 50 фраз. Фразы для анализа были взяты как из базы данных реальных запросов пользователей, так и из базы искусственно придуманных запросов для тестирования системы выделения семантического значения из предложения. Для каждой из взятых фраз используемая система не могла выделить семантическое значение, и все эти фразы были проанализированы специалистами, разрабатывающими эту систему, для определения слов и грамматических конструкций, которые препятствуют определению грамматического шаблона для них на этапе извлечения информации.

На основе этого анализа было сформулировано несколько общих правил, позволяющих исключать из предложения подобные конструкции, и при этом основывались бы на анализе синтаксического дерева, построенного синтаксическим анализатором. Применение каждого подобного правила к исходному предложению должно приводить к порождению новой

гипотезы, к которой, в свою очередь, опять будет применено это и другие правила, и так до тех пор, пока не будут сформулированы все гипотезы пользовательского ввода. Рассмотрим эти базовые правила.

1. **Пунктуация**. Самое первое правило, которое применяется еще до этапа синтаксического анализа предложения – это удаление знаков пунктуации и других специальных символов. Наличие подобных символов затрудняет синтаксический парсинг предложения и все равно не учитывается системами выделения смысла. Применение данного правила так же приводит все символы в предложении в нижний регистр.
2. Правило для удаления слова «**Number**». Из предложений выше видно, что подобное слово присутствует практически всегда перед номером чего-либо. Это правило можно считать тоже предварительной обработкой для удаления лишних слов, которые никак не влияют на выделение семантического значения, но препятствуют правильному составлению синтаксического дерева.

Подобный вывод был получен практическими экспериментами и, по решению специалистов, которые занимаются разработкой системы, это слово может быть без последствий удалено из предложения. Правило было разделено на две части. 1. Слова «num, num., numb, no, no., #, №», стоящие перед числительным, заменяются на слово «Number». 2. Слово «Number», стоящее перед числительным, удаляется из предложения. Пример: «describe witness **number (#,№, num)** 2» - эквивалентно «describe witness 2».

3. **Притяжательное окончание «s»**, Это правило было так же сформулировано по рекомендации лингвистов, ибо наличие данного окончания никак не влияет на определение семантического значения, но затрудняет построение синтаксического дерева. Пример: «l**et's** make our way to the witness, ».

Эти три правила применяются на этапе подготовки предложения и не приводят к формированию новой гипотезы пользовательского ввода, то есть просто изменяют исходное предложение. После применения этих «подготовительных» правил уже формируется

изначальное синтаксическое дерево, изменение которого будет приводить к созданию других гипотез пользовательского ввода. Однако стоит добавить, что применение данных правил должно приводить к уменьшению достоверности исходного предложения, так как эти правила все же изменяют его.

4. **Прилагательные перед существительными.** Удаление прилагательных, стоящих перед существительными, является основным правилом для избавления от описательных слов. Это правило самое простое и действенное, потому что именно такие прилагательные вызывают затруднения при выделении семантического значения из предложения и вызывают необходимость составления бесконечного числа грамматических шаблонов. Примеры необходимости вычеркивания таких слов: show me **quickest** way to witness, show me the **nearest** last month robberies, 1990s **green** jeep Cherokee, what's description for the **female** suspect и так далее. Дальнейшее исследование результатов работы алгоритма показывает, что данное правило позволяет добиться правильного распознавания

смысла предложения примерно в 50 процентах из тестового набора фраз, изначально не подходящих ни под один грамматический шаблон.

5. **Окончание «s»**, указывающее на период времени. Подобная частица, стоящая рядом с числительным, тоже удаляется из предложения. Это правило было так же сформулировано по рекомендации лингвистов, ибо наличие данной частицы не предусмотрено в существующих грамматических шаблонах и делает невозможным выделение семантического значения из фразы. Пример: «**1990s** green Ford».
6. **Удаление числительных.** Правило, по которому удаляются числительные из предложения. Синтаксический анализатор позволяет распознать числительные, написанные как цифрами, так и словами (first, second). Зачастую именно такие числительные не позволяют отнести фразу к какому-либо грамматическому шаблону. Пример: «describe **first** witness, get route to witness number **2**». Удаление этих числительных исправляет ситуацию.
7. **Удаление наречий.** Наречия тоже можно отнести к описательным словам, но лишь в тех случаях, когда

они несут в себе уточняющих смысл. Однако зачастую удаление наречия из предложения приводит к тому, что исходный смысл фразы меняется на противоположный. По этой причине было решено удалять лишь те наречия, которые стоят в предложении в сравнительной или превосходной степени. В чистой форме удаляются лишь те наречия, которые стоят перед другими наречиями. Наречия в чистой форме, стоящие перед любой другой частью речи остаются в предложении без изменения, что позволяет сохранить смысл сказанной фразы. Примеры: what area is today the **most** dangerous about murders, you speak **extremely** loudly (слово loudly останется – для сохранения исходного смысла).

8. **Удаление имен собственных.** Данное правило удаляет из предложения все имена собственные, что, конечно же, сильно снижает достоверность сформированной гипотезы, но в некоторых случаях просто необходимо. Пример таких фраз: Find me the quickest route to witness number 2 **Jake Verdinier**, take me to the residence of witness **Sally Heim**. Выделение семантического значения из подобных

фраз возможно только при исключении из них имен свидетелей, что, конечно же, является недостатком существующих грамматических шаблонов, но именно эту проблему и призвана решить моя работа.

9. **Удаление однородных и равнозначных конструкций.** Данное правило, в отличие от предыдущих, направлено на удаление сразу целых конструкций из предложений, а не отдельных слов. Его работу можно продемонстрировать на примере следующего синтаксического дерева: what's the best way to get to witness one from here.

```
(SBARQ
  (WHNP (WP what))
  (SQ (VBZ 's)
    (NP
      (NP (DT the) (JJS best) (NN way))
      (S
        (VP (TO to)
          (VP (VB get)
            (PP (TO to)
              (NP (NN witness) (CD one)))
            (PP (IN from)
              (NP (RB here)))))))))
```

Правило применяется при следующих условиях: 1) грамматические конструкции играют в предложении одинаковые синтаксические роли. 2) Грамматические конструкции (части дерева) имеют

одинаковую глубину и эта глубина больше единицы – это условие позволяет провести удаление именно грамматических конструкций, а не отдельных слов. Результатом работы данного правила будут две гипотезы: what's the best way to get **to witness one**, what's the best way to get **from here**. Стоит добавить, что вычисление достоверности сформированных гипотез будет производиться на основе порядка расположения удаляемых грамматических конструкций в предложении, так достоверность первой гипотезы будет выше, чем достоверность второй, что является решением поставленной задачи и отражает изначальный смысл предложения.

10. **Изменение предложных конструкций** (Preposition phrase). Такие конструкции обычно начинаются с предлогов about, in, with, from и других. В дереве, построенном синтаксическим анализатором, такие фразы легко выделить из предложения по наличию соединительного слова IN. Например:

```
(VP (VB tell)
  (NP (PRP me))
  (PP (IN about)
    (NP
```

(NP (NNS robberies))

(NP
(NP (NN show) (NN information))
(PP (IN about)
(NP (JJ second) (NN witness)))))

(SINV
(VP (VBZ is))
(NP
(NP (DT the)
(ADJP (JJ suspect) (VBN connected)))
(PP (IN with)
(NP (DT any) (RB gang) (NN activity))))))

Обработка таких конструкций будет заключаться в удалении из этих конструкций **всех** слов, кроме первого существительного. Например, из фразы is the suspect connected **with any gang activity** будет получена фраза is the suspect connected **with activity**. А из фразы give me all rapes **in the city that happened last week** - фраза give me all rapes **in the city**. Таким образом, данное правило помогает сформировать довольное интересные гипотезы без дополняющих конструкций, сократив их до одного ключевого слова.

11. **Удаление предложных конструкций внутри таких же предложных конструкций.** Примером работы этого правила будет следующее дерево:

```
(VP (VB find)
  (NP (PRP$ my) (NN video) (NN record))
  (PP (IN with)
    (NP
      (NP (NN car))
      (VP (VBG crashing)
        (PP (IN in)
          (NP (DT the) (NN gallery)))))))))
```

Видно, что одна предложная конструкция вложена в другую, что, скорее всего, говорит о ее описательном значении. Применив данное правило можно получить следующую гипотезу: find my video record with car crashing. Или из фразы tell about a pink Jeep **with leather seats** – гипотезу tell about a pink Jeep, что является прекрасным примером полезности этого правила.

12. **Удаление предложных конструкций.** Это правило подразумевает полное удаление предложных конструкций из предложения, что, конечно же, требуют значительного уменьшения достоверности таких гипотез, но в некоторых ситуациях просто необходимо. Пример такой фразы: white ford pickup **with license plate ABC123**, blue Chevy sedan **with Arizona license abc123**. Выделение семантического

значения из таких фраз возможно только после применения данного правила.

2.4.Применение правил

При применении сформулированных выше правил семантического сокращения предложений нужно учитывать несколько особенностей. Рассмотрим каждую из них.

Первое. Каждое правило работает путем рекурсивного спуска по синтаксическому дереву и одновременному анализу одного конкретного уровня дерева. Если на этом уровне можно применить правило, то оно применяется, и каждое такое применение приводит к формированию новой гипотезы, причем сформированная гипотеза записывается, но ее анализ не происходит. Если правило нельзя применить на этом уровне, алгоритм спускается ниже. После прохождения всех уровней исходного дерева, сформированные гипотезы добавляются в общий список гипотез и запоминаются. Далее происходит применение этого правила для только что сформированных гипотез. Данный процесс происходит до тех пор, пока применение правила приводит к формированию новых

гипотез. Рассмотрим эту особенность на примере работы правила INremovingRule (удаление предложных конструкций) над следующим деревом:

```
(VP (VB tell)
  (NP (PRP me))
  (PP (IN about)
    (NP
      (NP (NNS robberies))
      (PP (IN in)
        (NP (CD 2) (NNS miles))))))))
```

Анализ дерева происходит рекурсивно «сверху вниз», поэтому первое срабатывание правила приведет к формированию гипотезы «tell me», после чего анализ продолжится и будет сформирована еще одна гипотеза – «tell me about robberies». После этого применение правила будет закончено и к списку гипотез будет добавлено две новые - «tell me» и «tell me about robberies». Так как правилу удалось сформировать две гипотезы из исходной фразы, для каждой из них будет произведено повторное применение этого правила, что приведет к формированию третье гипотезы - «tell me». На этом применение правила будет закончено. Результат – три гипотезы: «tell me», «tell me about robberies» и «tell me».

Отсюда возникает вторая особенность применения подобных правил – после срабатывания каждого из них необходимо исключить из списка гипотез повторяющиеся. Так как правило INremovingRule сформировало три гипотезы, но две из них повторяются, то результатом работы данного правила будет только две конечные гипотезы - «tell me» и «tell me about robberies», третья будет удалена как повторяющаяся.

2.5.Восстановление предложения

Основой разрабатываемого модуля формирования гипотез пользовательского ввода являются описанные выше правила работы с синтаксическими деревьями. Эти правила приводят к изменению синтаксического дерева, которые было сформировано синтаксическим анализатором, и результатом работы этих правил являются точно такие же деревья – с измененными или удаленными листьями или ветками. Но в качестве результата работы всего модуля мы должны получить именно предложения, поэтому необходимо произвести обратное преобразование синтаксического дерева (измененного

правилами) в предложение (гипотезу). К сожалению, библиотека The Stanford Parser, используемая в проекте, не предоставляет стандартных инструментов для этого.

Существенной особенностью составления синтаксических деревьев с помощью этой библиотеки является то, что порядок следования листьев в таких деревьях соответствует порядку слов в исходном предложении. Например, для предложения «Describe second witness in case 1112» будет сформировано уже знакомое нам дерево:

```
(S
  (VP (VB Describe)
    (NP (JJ second) (NN witness))
    (PP (IN in)
      (NP (NN case) (CD 1112))))))
```

Данное дерево хранится в памяти в виде экземпляра класса Tree используемой библиотеки, у которого есть стандартный метод getLeaves(), возвращающий массив листьев данного дерева. Данный метод вернет массив листьев этого дерева следующего вида:

(VB Describe)(JJ second)(NN witness)(IN in)(NN case)(CD 1112)

И даже изменение исходного дерева не приведет к нарушению порядка следования этих листьев. Таким образом, для того, чтобы из синтаксического дерева восстановить предложения, достаточно получить список листьев этого дерева и извлечь из этого списка слова (Leave.getWord()). Именно таким образом в модуле формирования гипотез будет происходить восстановления предложений после формирования гипотез.

2.6.Общая архитектура проекта

После описания всех ключевых этапов работы алгоритма можно сформировать общую архитектуру проекта. Изобразим ее в виде схемы:

Рис. 4 Последовательность формирования гипотез ввода

3. Программная реализация модуля составления гипотез

Как было сказано ранее, метод генерации гипотез пользовательского ввода основан на анализе синтаксического дерева исходного предложение. Такое синтаксическое дерево было решено строить средствами библиотеки **Stanford CoreNLP**, а именно при помощи синтаксического анализатора предложений The Stanford Parser, входящего в эту библиотеку. Код данной библиотеки написан на языке высокого уровня Java и доступен под лицензией GPL для использования в некоммерческих проектах. Так же существуют решения для подключения библиотеки к другим языкам, но было решено не использовать их, а реализовывать модуль генерации гипотез на языке Java.

Программная реализация проекта будет происходить в среде разработки IntelliJ IDEA the Java IDE от компании JetBrains, сборка проекта будет осуществляться инструментом для сборки Java проектов – Maven.

Для использования библиотеки Stanford CoreNLP в проекте достаточно добавить зависимости в файл описания сборки Maven – **pom.xml**. Файлы библиотеки автоматически загрузятся с официального сайта и станут доступны для использования в проекте.

```
<dependency>
   <groupId>edu.stanford.nlp</groupId>
   <artifactId>stanford-corenlp</artifactId>
   <version>3.6.0</version>
</dependency>

<dependency>
   <groupId>edu.stanford.nlp</groupId>
   <artifactId>stanford-corenlp</artifactId>
   <version>3.6.0</version>
   <classifier>models</classifier>
</dependency>
```

3.1. Основной класс модуля

Основной класс модуля HypGenerator является входной точкой для всего алгоритма. В конструкторе этого класса происходит инициализация средств библиотеки Stanford CoreNLP, инициализация классов всех правил и вспомогательных инструментов.

Помимо конструктора класс содержит метод generateHypothesis(String inputSentence), принимающий в качестве входного параметра исходное предложение

пользователя и возвращающий список из гипотез, построенных на основе этого предложения. Метод генерации гипотез последовательно выполняет следующие операции:

1. Применение к исходному предложению правил первичной обработки: PunctuationRule, SProcessingRule, NumberProcessingRule.
2. Формирование на основе обработанного предложения исходной (оригинальной) гипотезы средствами синтаксического анализатора и добавление ее в список сформированных гипотез – resultList. Достоверность исходной гипотезы считается равной единице или задается входным параметром.
3. Последовательное применение к списку сформированных гипотез всех правил семантического сокращения предложения и обновление списка гипотез после применения каждого правила.
4. Удаление из списка гипотез повторяющихся предложений после применения каждого из правил семантического сокращения.
5. Сортировка списка гипотез по убыванию значения достоверности каждой из гипотез.

6. Возвращение списка сформированных гипотез пользовательского ввода.

Программный код реализации основного класса модуля генерации гипотез пользовательского ввода приведен в Приложении №1 данной работы.

3.2.Класс гипотезы пользовательского ввода

Все методы и классы реализуемого модуля оперируют объектами элементарного класса InputHypothesis. Этот класс описывает гипотезу пользовательского ввода и содержит два поля: **hTree** – синтаксическое дерево гипотезы и **hConfidence** – экземпляр класса HypothesisConfidence, описывающего достоверность гипотезы и содержащий методы для ее изменения и обновления.

Конструктор класса InputHypothesis принимает два параметра – синтаксическое дерево (экземпляр класса Tree библиотеки Stanford CoreNLP) и исходное значение достоверности гипотезы (либо экземпляр класса HypothesisConfidence).

Помимо этого реализован метод для сравнения объектов данного класса для сортировки списка

сформированных гипотез. Сравнение объектов происходит путем сравнения значений достоверностей этих гипотез.

Класс, описывающий достоверность гипотезы, - **HypothesisConfidence** содержит в себе три поля – число слов в гипотезе (wordCount), глубина синтаксического дерева (treeDeep), значение достоверности (confidence). Каждое из этих полей используется для расчета нового значения достоверности в методе этого класса updateConfidence.

Программный код реализации этих классов также приведен в Приложении №2 и в Приложении № 3 данной работы.

3.3.Реализация правил семантического сокращения

В разделе «Разработка метода генерации гипотез пользовательского ввода» было сформулировано 12 правил семантического сокращения предложений, которые формируют новые гипотезы пользовательского ввода. Для того чтобы алгоритм был легко расширяем и была возможность простого добавления новых правил в

систему – было решено каждое правило реализовывать в виде отдельного класса. Помимо этого все классы, описывающие программную реализацию правил, наследуются от общего предка – класса **BaseHypothesisRule**. Такая структура проекта позволяет привести объект любого класса из правил к общему предку и хранить все экземпляры этих классов в одном массиве (смотри реализацию класса HypGenerator).

Класс BaseHypothesisRule содержит несколько методов:

1. getRuleName() – возвращает название правила. Используется для отладки и выведения информации на экран.
2. cleanHypothesisList(List<InputHypothesis>) – метод удаляет из заданного списка гипотез повторяющиеся предложения. Определение повторов происходит путем преобразования синтаксического дерева каждой гипотезы в строку и сравнения этих строк между собой. В случае обнаружения повтора – удаляется гипотеза с **большим** значением достоверности.

3. getNewTree(Tree) – метод формирует новое дерево на основе исходного. Подобная процедура необходима по той причине, что после вычеркивания грамматических конструкций из дерева оно теряет свою семантическую целостность, и дальнейшее применение других правил не приносит никакого результата. Поэтому после формирования каждой гипотезы в качестве ее синтаксического дерева используется вновь сформированное этим методом дерево.
4. getHypothesis(List<InputHypothesis>) – основной функциональный метод для каждого правила. Его назначение – из списка входных гипотез сформировать новые и вернуть их в основной класс модуля. Данный метод переопределен в каждом наследующем классе с учетом правила семантического сокращения, которое реализует этот класс.

В свою очередь каждый из 13 классов, реализующих функционал того или иного правила семантического сокращения предложения, является потомком класса BaseHypothesisRule, и в своей реализации переопределяет метод getHypothesis.

3.4.Вспомогательные и служебные классы

Класс **CoreNlpPipeline** – является интерфейсом для удобной работы с методами библиотеки Stanford CoreNLP. В этом классе происходит инициализация этой библиотеки и написаны методы для обращения к ней, например методы формирования дерева из строки, содержащей предложение, или метод для формирования сразу списка деревьев, если в строке содержится несколько отдельных предложений. Создание подобного класса было вызвано необходимостью выполнения нескольких обращений к библиотеке для формирования синтаксического дерева, что было бы неудобно делать в основном цикле программы.

Класс **CoreNlpOutput** используется для обратной процедуры - восстановления предложения по синтаксическому дереву. Алгоритм восстановления предложения на основе дерева был описан в разделе «Разработка метода генерации гипотез пользовательского ввода». Так же в классе реализованы методы для вывода синтаксического дерева на экран, вывода на экран списка гипотез, которые используются на этапе отладки.

Класс **CoreNlpConstants** содержит в себе строковое описание констант, используемых в обозначениях грамматических конструкций и различных частей речи в синтаксических деревьях библиотеки Stanford CoreNLP. Так же в этом классе определены коэффициенты, которые используются для расчета значений достоверности гипотез, подробное описание и назначение этих констант приведено в следующей главе.

4. Разработка метода оценки достоверности гипотезы

Любая система извлечения информации из произнесенного пользователем предложения принимает на вход список из наборов «гипотеза - ее достоверность». Подобный список гипотез и их достоверностей формируется программами распознавания человеческой речи и перевода ее в текст. Необходимость формирования нескольких гипотез произнесенной фразы вызвана тем, что системы распознавания речи работают, чаще всего, по принципу машинного обучения или на основе обучаемых нейронных сетей, а такие алгоритмы зачастую не могут дать однозначного ответа на вопрос – что сказал пользователь. Поэтому обычная схема работы голосовых помощников заключается в том, что на вход модуля выделения смысловой составляющей поступает набор гипотез с системы распознавания речи. При реализации моего проекта, а именно модуля формирования гипотез пользовательского ввода, было решено воспользоваться этой особенностью и «встроить» данный алгоритм между системой

распознавания речи и модулем извлечения семантического значения. Подобная интеграция заключается в расширении списка гипотез искусственно сформированными гипотезами. Так же было решено добавить в алгоритм формирования гипотез правило расчета достоверности искусственно сформированных гипотез, для того, чтобы модуль извлечения смысла мог оценить, насколько та или иная гипотеза соответствует изначальному предложению, произнесенному пользователем.

Расчет достоверности для сформированных гипотез должен опираться на несколько основных принципов:

1. Достоверность вновь сформированной гипотезы не может быть ниже достоверности исходной гипотезы.
2. Достоверность гипотез должна оцениваться показателем от 0.0 до 1.0. Этого требуют все системы выделения смысловой составляющей из предложения.
3. Расчет достоверности гипотезы должен происходить с учетом количества вычеркнутых из

предложения слов. Чем больше слов вычёркивается при применении того или иного правила – тем ниже достоверность этой гипотезы.

4. Расчет достоверности гипотезы должен происходить с учетом синтаксической роли вычеркнутых слов из предложения, или, проще говоря, должен зависеть от части речи вычеркиваемого слова.
5. Расчет достоверности гипотезы должен происходить с учетом особенности применяемого правила семантического сокращения и степени влияния того или иного правила на общее семантическое значение после его применения.

4.1. Программная реализация модуля оценки достоверности гипотезы

Как уже было сказано выше, оценка достоверности гипотезы происходит при каждом формировании тем или иным правилом новых гипотез. При этом новое значение достоверности гипотезы рассчитывается на основе достоверности исходной гипотезы. Так, например, если сказанная пользователем фраза имела достоверность равную 1.0, то сформированная на основе этой фразы гипотеза будет иметь достоверность, рассчитанную на основе этого значения и меньше его. Например, сформированная гипотеза получила значение достоверности равное 0.8, в этом случае следующая гипотеза, производная от нее, будет иметь достоверность, рассчитанную на основе этого значения и опять же ниже его.

Для того чтобы реализовать данный механизм расчета достоверности, было решено не создавать отдельный модуль для этого действия, а производить расчет нового значения внутри класса InputHypothesis, в котором, как уже было описано выше, значение

достоверности содержится в поле hConfidence (экземпляр класса HypothesisConfidence).

Таким образом, формирование новой гипотезы и оценка ее достоверности в разрабатываемом модуле происходит следующим способом:

1. Начало применения правила семантического сокращения для исходной гипотезы.
2. В случае если возможно применить правило и сформировать новую гипотезу, происходит копирование исходной гипотезы (копирование ее дерева и объекта HypothesisConfidence). Алгоритм правила производит изменение синтаксического дерева для копии исходной гипотезы, тем самым формируя новую гипотезу пользовательского ввода.
3. После изменения дерева, алгоритм применения правила вызывает метод updateConfidence класса HypothesisConfidence для новой гипотезы. Входными параметрами этого метода являются: количество удаленных из дерева слов, их части речи, название правила, совершившего это сокращение.

4. На основе количества удаленных слов, их частей речи и названия правила семантического сокращения в методе updateConfidence происходит пересчет значения достоверности новой гипотезы. Помимо этих параметров в формуле расчета новой достоверности используются: количество слов в гипотезе до удаления, значение достоверности гипотезы до удаления из нее слов. Эти значения, как было сказано выше, хранятся в классе HypothesisConfidence.

Расчет достоверности новой гипотезы происходит по следующей формуле:

$$hC = hC - hC * \left(\frac{rWc}{tWc}\right) * POSc * Rc;$$

Где hC – значение достоверности гипотезы, rWc – количество удаленных слов, tWc – количество слов в гипотезе до удаления, POSc – коэффициент для каждой части речи, Rc – коэффициент для применяемого правила. Коэффициенты для частей речи и применяемых правил определены в классе CoreNlpConstants и могут быть изменены для регулирования правильности расчета достоверности.

Следует заметить две особенности расчета значения достоверности:

1. Если при применении правил семантического сокращения предложения были удалены слова разных частей речи, то метод updateConfidence вызывается несколько раз – по разу для каждой из частей речи с разными значениями коэффициентов и количества удаленных слов.
2. Обновление общего количества слов в гипотезе происходит после полного пересчёта значения достоверности для новой гипотезы.

Применение данного способа расчета значения достоверности позволяет добиться реализации всех требований, предъявляемых к методу формирования достоверности гипотезы. А возможность изменения коэффициентов расчета достоверности для разных частей речи и применяемых правил позволят производить корректировку работы алгоритма при формировании и сортировки произведенных гипотез пользовательского ввода.

5. Анализ результатов работы

Оценка результатов работы разработанного алгоритма была проведена по двум критериям: первый – процентное соотношение количества сформированных ожидаемых гипотез ввода к общему числу ожидаемых (необходимых) гипотез на основе исходного набора тестовых фраз, второй – оценка количества фраз из тестового набора, для которых удалось сформировать ожидаемые гипотезы ввода. Таким образом, можно будет оценить и качество работы алгоритма (на основе первого критерия) и его практическую полезность (второй критерий).

5.1. Демонстрация работы модуля и тестирование

Перед разработкой правил семантического сокращения предложения был создан тестовый набор фраз реальных пользователей системы, выделение смысловой составляющей из которых невозможно по причине излишней распространенности этих предложений, что мешало отнести их к какому-либо существующему грамматическому шаблону. Именно на основе этих фраз и желаемых гипотез из них были

разработаны правила семантического сокращения, которые и производят формирование этих желаемых гипотез. На данном этапе работы хотелось бы привести примеры применения алгоритма к этим фразам и список гипотез, которые удалось сформировать из них. Ниже приведены результаты работы программы в следующем виде: Исходное предложение, ожидаемые гипотезы, список гипотез, полученных после применения всех правил, процент совпадения требуемых гипотез и сформированных.

Пример 1. I need to get to witness 3 in case 8776
Ожидаемые гипотезы:

- I need to get to witness in case 8776
- I need to get to witness 3
- I need to get to witness in case
- I need to get to witness

Результат работы алгоритма:
Input:
0. I need to get to witness 3 in case 8776

Result:
1. w:10 d:11 c:1.0 : I need to get to witness 3 in case 8776
2. w:9 d:11 c:0.944 : I need to get to witness in case 8776
3. w:9 d:11 c:0.93 : I need to get to witness 3 in case
4. w:8 d:11 c:0.8705777777777778 : I need to get to witness in case
5. w:7 d:9 c:0.80352 : I need to get to witness 3
6. w:6 d:9 c:0.7394537174211248 : I need to get to witness
Сформированные гипотезы: **4 из 4**

Пример 2. let make our way to the residence of witness number 2

Ожидаемые гипотезы:

- let make way to the residence of witness 2
- let make way to the residence of witness
- let make way to witness
- let make way to the residence

Результат работы алгоритма:

Input:

0. let make our way to the residence of witness number 2

Result:

1. w:10 d:9 c:0.9818181818181818 : let make our way to the residence of witness 2
2. w:9 d:9 c:0.9425454545454546 : let make way to the residence of witness 2
3. w:9 d:9 c:0.9130909090909091 : let make our way to the residence of witness
4. w:8 d:9 c:0.8692363636363637 : let make way to the residence of witness
5. w:7 d:7 c:0.7889105454545454 : let make our way to the residence
6. w:6 d:7 c:0.7383143434343434 : let make way to the residence

Сформированные гипотезы: **3 из 4**

Пример 3. show the route to the main eyewitness for incident 7865

Ожидаемые гипотезы:

- show the route to the eyewitness for incident 7865
- show the route to the eyewitness for incident
- show the route to the main eyewitness
- show the route to the eyewitness

Результат работы алгоритма:

Input:

0. show the route to the main eyewitness for incident 7865

Result:

1. w:10 d:8 c:1.0 : show the route to the main eyewitness for incident 7865
2. w:9 d:8 c:0.93 : show the route to the main eyewitness for incident
3. w:9 d:8 c:0.91 : show the route to the eyewitness for incident 7865
4. w:8 d:8 c:0.8392222222222223 : show the route to the eyewitness for incident
5. w:7 d:6 c:0.80352 : show the route to the main eyewitness
6. w:6 d:6 c:0.7128208504801098 : show the route to the eyewitness

Сформированные гипотезы: **4 из 4**

Пример 4. describe the two male suspects in incident 769795

Ожидаемые гипотезы:

- describe the two suspects in incident 769795
- describe the suspects in incident 769795
- describe the suspects in incident
- describe the suspects

Результат работы алгоритма:

Input:
0. describe the two male suspects in incident 769795

Result:
1. w:8 d:7 c:1.0 : describe the two male suspects in incident 769795
2. w:7 d:7 c:0.93 : describe the male suspects in incident 769795
3. w:7 d:6 c:0.93 : describe the two male suspects in incident
4. w:7 d:7 c:0.8875 : describe the two suspects in incident 769795
5. w:6 d:6 c:0.8556 : describe the male suspects in incident
6. w:6 d:5 c:0.8312499999999999 : describe the two male suspects 769795
7. w:6 d:6 c:0.8165 : describe the suspects in incident 769795
8. w:6 d:6 c:0.8165 : describe the two suspects in incident
9. w:5 d:5 c:0.7515918367346939 : describe the male suspects 769795
10. w:5 d:5 c:0.7515918367346939 : describe the two male suspects
11. w:5 d:6 c:0.7402933333333334 : describe the suspects in incident
12. w:5 d:5 c:0.7172448979591837 : describe the two suspects 769795
13. w:4 d:4 c:0.6654666666666667 : describe the male suspects
14. w:4 d:5 c:0.6350555555555556 : describe the suspects 769795
15. w:4 d:5 c:0.6350555555555556 : describe the two suspects
16. w:3 d:4 c:0.5448558933333334 : describe the suspects

Сформированные гипотезы: **4 из 4**

Пример 5. find my video record with smth in gallery
Ожидаемые гипотезы:

- find my video record with smth
- find my video record in gallery
- find video record with smth in gallery
- find video record with smth

Результат работы алгоритма:
Input:
0. find my video record with smth in gallery

Result:
1. w:8 d:8 c:1.0 : find my video record with smth in gallery
2. w:7 d:8 c:0.95 : find video record with smth in gallery
3. w:6 d:6 c:0.8312499999999999 : find my video record with smth
4. w:5 d:6 c:0.7677551020408163 : find video record with smth
5. w:4 d:5 c:0.6909765625 : find my video record
6. w:3 d:5 c:0.6204714702207413 : find video record
Сформированные гипотезы: **4 из 4**

Как видно из перечисленных выше примеров – алгоритм успешно справляется со своей задачей и формирует все ожидаемые гипотезы. Но данные примеры я бы назвал «искусственными», все эти предложения сформулированы изначально таким образом, чтобы было удобно применять разрабатываемый алгоритм. К тому же правила семантического сокращения, используемые в алгоритме, разрабатывались именно с учетом этих примеров. Ниже

я хотел бы привести еще 5 запросов, которые были получены из базы обращений реальных пользователей (полицейских), а требуемые гипотезы были сформулированы лингвистами – разработчиками реально существующего голосового помощника.

Пример 6. blue 2007 Chevy Silverado from Nebraska commercial 68-113
Ожидаемые гипотезы:

- blue Chevy Silverado
- blue 2007 Chevy Silverado
- Chevy Silverado
- 2007 Chevy Silverado

Результат работы алгоритма:
Input:
0. blue 2007 Chevy Silverado from Nebraska commercial 68-113

Result:
1. w:8 d:6 c:1.0 : blue 2007 Chevy Silverado from Nebraska commercial 68-113
2. w:7 d:6 c:0.93 : blue Chevy Silverado from Nebraska commercial 68-113
3. w:7 d:6 c:0.93 : blue 2007 Chevy Silverado from Nebraska commercial
4. w:6 d:6 c:0.874158984375 : blue 2007 Chevy Silverado from Nebraska
5. w:6 d:6 c:0.8556 : blue Chevy Silverado from Nebraska commercial
6. w:6 d:4 c:0.8371875 : blue 2007 Chevy Silverado commercial 68-113
7. w:6 d:6 c:0.8310141927083333 : Chevy Silverado from Nebraska commercial 68-113
8. w:5 d:4 c:0.7578551020408164 : blue Chevy Silverado commercial 68-113
9. w:5 d:6 c:0.7130000000000001 : blue Chevy Silverado from Nebraska

10. w:4 d:4 c:0.6528078158275463 : Chevy Silverado commercial 68-113
11. w:4 d:4 c:0.649590087463557 : blue 2007 Chevy Silverado
12. w:5 d:5 c:0.6410569196428572 : blue 2007 from commercial 68-113
13. w:3 d:4 c:0.5601011111111112 : blue Chevy Silverado

Сформированные гипотезы: **2 из 4**

Пример 7. red 2010 Ford Mustang with broken rear window and rims

Ожидаемые гипотезы:

- red ford mustang
- red 2010 ford mustang
- ford mustang

Результат работы алгоритма:

Input:
0. red 2010 Ford Mustang with broken rear window and rims

Result:
1. w:10 d:7 c:1.0 : red 2010 Ford Mustang with broken rear window and rims
2. w:9 d:7 c:0.944 : red Ford Mustang with broken rear window and rims
3. w:9 d:9 c:0.935 : red 2010 Ford Mustang with broken rear window and
4. w:9 d:6 c:0.91 : red 2010 Ford Mustang with rear window and rims
5. w:8 d:9 c:0.8758222222222222 : red Ford Mustang with broken rear window and
6. w:8 d:6 c:0.8533777777777778 : red Ford Mustang with rear window and rims
7. w:8 d:6 c:0.8190000000000001 : red 2010 Ford Mustang with window and rims
8. w:8 d:6 c:0.8094722222222223 : red 2010 with broken rear window and rims
9. w:7 d:6 c:0.7650607902777777 : red 2010 Ford Mustang with and rims
10. w:7 d:6 c:0.7616700000000001 : red Ford Mustang with window and rims

11. w:7 d:6 c:0.7440883333333334 : red with broken rear window and rims
12. w:7 d:8 c:0.7437026041666667 : red 2010 with broken rear window and
13. w:6 d:5 c:0.7213430308333333 : red 2010 Ford Mustang and rims
14. w:6 d:6 c:0.7186725 : red 2010 Ford Mustang with rear
15. w:7 d:5 c:0.7172885416666667 : red 2010 with rear window and rims
16. w:6 d:6 c:0.7077777777777778 : red 2010 Ford Mustang with window
17. w:6 d:6 c:0.6999684624599908 : red Ford Mustang with and rims
18. w:6 d:8 c:0.6749944166666667 : red with broken rear window and
19. w:5 d:4 c:0.6533038982959913 : red Ford Mustang and rims
20. w:5 d:6 c:0.6526419753086419 : red Ford Mustang with rear
21. w:6 d:5 c:0.6499767083333334 : red with rear window and rims
22. w:5 d:6 c:0.6419917686631944 : red Ford Mustang with window
23. w:6 d:5 c:0.6237928125000001 : red 2010 with window and rims
24. w:4 d:4 c:0.6209330399999999 : red 2010 Ford Mustang
25. w:5 d:6 c:0.5769248204774456 : red 2010 with and rims
26. w:3 d:4 c:0.5543428135954884 : red Ford Mustang

Сформированные гипотезы: **2 из 3**

Пример 8. a 2015, blue hyundai sonata occupied 2 times

Ожидаемые гипотезы:

- a blue hyundai sonata
- a hyundai sonata
- blue hyundai
- a 2015 hyundai sonata

Результат работы алгоритма:

Input:
0. A 2015, blue hyundai sonata occupied 2 times

Result:
1. w:9 d:8 c:1.0 : A 2015 blue hyundai sonata occupied 2 times
2. w:7 d:7 c:0.9377777777777778 : A blue hyundai sonata occupied 2 times
3. w:7 d:4 c:0.9377777777777778 : A 2015 blue hyundai sonata occupied times
4. w:7 d:7 c:0.9 : A 2015 hyundai sonata occupied 2 times

5. w:6 d:4 c:0.8627555555555556 : A blue hyundai sonata occupied times
6. w:6 d:7 c:0.8280000000000001 : A hyundai sonata occupied 2 times
7. w:6 d:4 c:0.8280000000000001 : A 2015 hyundai sonata occupied times
8. w:5 d:4 c:0.7796043310657597 : A 2015 blue hyundai sonata
9. w:5 d:4 c:0.75072 : A hyundai sonata occupied times
10. w:5 d:4 c:0.7222000000000001 : A 2015 hyundai occupied times
11. w:4 d:4 c:0.6942845592592592 : A blue hyundai sonata
12. w:4 d:4 c:0.6782557680272109 : A 2015 blue hyundai
13. w:4 d:4 c:0.6663157500000001 : A 2015 hyundai sonata
14. w:3 d:5 c:0.6146843494290124 : sonata occupied times
15. w:2 d:3 c:0.5844228005947586 : occupied times
16. w:3 d:4 c:0.5814633183796296 : A blue hyundai
17. w:3 d:3 c:0.5767106112 : A hyundai sonata
18. w:2 d:3 c:0.5563938721181532 : blue hyundai
19. w:3 d:4 c:0.554801262 : A 2015 hyundai

Сформированные гипотезы: **4 из 4**

Пример 9. suspicious vehicle is a red 2015 Scion FR-S, 2 door, Washington Plate AXM1254

Ожидаемые гипотезы:

- vehicle is a red 2015 scion fr-s 2 door
- vehicle is a red 2015 scion fr-s
- red 2015 scion fr-s
- red 2015 scion

Результат работы алгоритма:

Input:

0. suspicious vehicle is a red 2015 Scion FR-S, 2 door, Washington Plate AXM1254

Result:

1. w:15 d:6 c:1.0 : suspicious vehicle is a red 2015 Scion FR-S 2 door Washington Plate AXM1254
2. w:12 d:5 c:0.9626666666666667 : suspicious vehicle is a red Scion FR-S 2 door Washington Plate AXM1254
3. w:12 d:7 c:0.9626666666666667 : suspicious vehicle is a red 2015 Scion FR-S door Washington Plate AXM1254
4. w:12 d:7 c:0.94 : vehicle is a red 2015 Scion FR-S 2 door Washington Plate AXM1254

5. w:11 d:5 c:0.9177422222222222 : suspicious vehicle is a red Scion FR-S door Washington Plate AXM1254
6. w:11 d:7 c:0.9105222222222222 : suspicious vehicle is a red 2015 Scion FR-S door Washington Plate
7. w:11 d:7 c:0.9086175000000001 : suspicious vehicle is a red 2015 Scion FR-S Washington Plate AXM1254
8. w:11 d:5 c:0.8961333333333332 : vehicle is a red Scion FR-S 2 door Washington Plate AXM1254
9. w:11 d:8 c:0.8961333333333332 : vehicle is a red 2015 Scion FR-S door Washington Plate AXM1254
10. w:10 d:7 c:0.8795223687962963 : suspicious vehicle is a red 2015 Scion FR-S 2 door
11. w:11 d:7 c:0.8731071428571429 : suspicious vehicle is a red 2015 Scion FR-S 2 door AXM1254
12. w:10 d:5 c:0.8635120000000001 : suspicious vehicle is a red Scion FR-S door Washington Plate
13. w:10 d:5 c:0.8549264659090909 : suspicious vehicle is a red 2015 Scion FR-S Washington Plate
14. w:10 d:6 c:0.8505119999999999 : vehicle is a red Scion FR-S door Washington Plate AXM1254
15. w:10 d:6 c:0.8332475781249999 : vehicle is a red 2015 Scion FR-S Washington Plate AXM1254
16. w:10 d:5 c:0.8098980303030303 : suspicious vehicle is a red Scion FR-S 2 door AXM1254
17. w:10 d:5 c:0.8098980303030303 : suspicious vehicle is a red 2015 Scion FR-S door AXM1254
18. w:9 d:7 c:0.7999360974428529 : vehicle is a red 2015 Scion FR-S 2 door
19. w:10 d:7 c:0.790828409090909 : vehicle is a red 2015 Scion FR-S 2 door AXM1254
20. w:9 d:5 c:0.7837521597773439 : suspicious vehicle is a red 2015 Scion FR-S AXM1254
21. w:9 d:5 c:0.7588268133333333 : suspicious vehicle is a red Scion FR-S door AXM1254
22. w:9 d:5 c:0.7409596999999999 : vehicle is a red Scion FR-S 2 door AXM1254
23. w:9 d:5 c:0.7409596999999999 : vehicle is a red 2015 Scion FR-S door AXM1254
24. w:7 d:5 c:0.700050229891155 : vehicle is a red Scion FR-S door
25. w:8 d:5 c:0.6884658386666666 : vehicle is a red Scion FR-S door AXM1254

26. w:7 d:7 c:0.6858400100441315 : vehicle is a red 2015 Scion FR-S
Сформированные гипотезы: **2 из 4**

Таким образом, можно сделать вывод, что разрабатываемый метод формирования гипотез пользовательского ввода практически всегда справляется со своей задачей. На тестовых фразах, которые использовались для составления правил семантического значения – процент генерации ожидаемых гипотез близок к 100, на реальных запросах пользователей алгоритму удается сформировать порядка 70 процентов ожидаемых гипотез.

Помимо возможностей по формированию ожидаемых гипотез из исходных фраз пользователей (так называемый **true positive rate**) было проведено исследование алгоритма на вероятность формирования нежелательных гипотез из фраз, где подобные гипотезы не должны были быть сформированы. **Например:**

Фраза: find Sheldon Cooper. Нежелательная гипотеза: find Cooper. Подобная гипотеза приведет к тому, что будет неверное распознано намерение пользователя – вместо поиска человека будет найдена машина.

Фраза: he was the challenger on the game show. Нежелательная гипотеза: Challenger.

Фраза: we went on vacation and saw the wild mustang. Нежелательная гипотеза: saw the mustang.

5.2. Результаты тестирования

В ходе первого этапа тестирования с помощью алгоритма были сформированы гипотезы для 258 высказываний. 50 из них были искусственно созданы для разработки алгоритма, остальные 208 взяты из реальных запросов пользователей. Для фраз из реальных запросов пользователей было сформулировано 83 **ожидаемых** гипотез и 198 нежелательных гипотез. Результаты тестирования и график приведены ниже.

Из графика, отображающего результаты тестирования, видно, что алгоритму удалось успешно сформировать 96 процентов ожидаемых гипотез для искусственно сформированных запросов и порядка 65 процентов ожидаемых гипотез для реальных запросов пользователей. При этом из нежелательных гипотез было сформировано только лишь 2 процента гипотез.

Таблица 2 Результаты тестирования алгоритма

Тестовый набор	Количество ожидаемых гипотез	Удалось сгенерировать	Кол-во фраз, для которых сработал алгоритм
Искусственные фразы	195	188	49 из 50
Реальные фразы (positive test)	83	53	23 из 26
Реальные фразы (negative test)	198	3	3 из 182

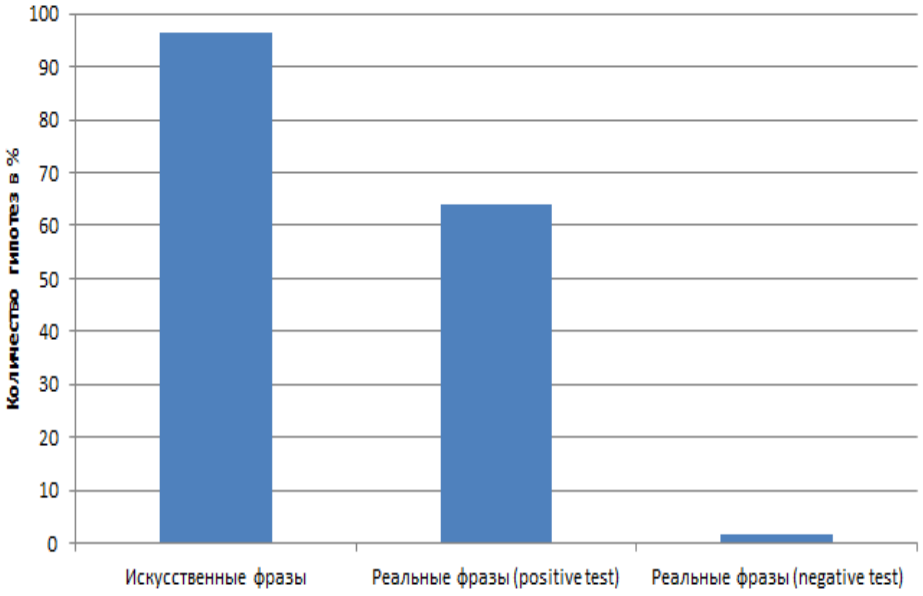

Рис. 5 Результаты тестирования алгоритма

Результаты тестирования показывают, что алгоритм формирует ожидаемые гипотезы для 72 тестовых фраз из 76, что говорит об отличных результатах моей работы. Формирования ожидаемых гипотез для всех этих фраз означает, что для каждой из них система выделения смысловой составляющей, способна распознать намерение пользователя, хотя изначально все эти фразы было невозможно обработать подобной системой. Таким образом, предложенный в работе метод увеличивает вероятность выделения смысловой составляющей для изначально неподходящих ни под один грамматический шаблон фраз без составления новых грамматических шаблонов или расширения существующих словарей.

Заключение

В ходе выполнения данной работы мной были изучены основные технологии, которые используются в такой области информационных технологий, как понимание естественной речи. Были рассмотрены основные принципы работы систем извлечения информации из предложений на естественном языке, проблемы, возникающие при создании этих систем и основные способы решения их.

В рамках работы был разработан метод генерации гипотез пользовательского ввода на основе синтаксического анализа исходного выражения пользователя для решения проблемы излишней распространенности предложений при работе систем выделения намерений пользователя, основанных на описании грамматических шаблонов языка. Была создана программная реализация этого метода и определены критерии тестирования разработанного алгоритма.

Результаты тестирования модуля показывают, что разработанный алгоритм успешно справляется со

своей задачей. С помощью алгоритма удалось сформировать большую часть гипотез пользовательского ввода, которые позволили бы выделить смысловую составляющую сказанной пользователем фразы, хотя до применения алгоритма данные предложения не подходили ни под один грамматический шаблон.

Интеграция данного алгоритма в существующую систему понимания естественной речи позволила бы существенно повысить точность работы такой системы и успешно определить намерения пользователя без необходимости изменения существующих грамматических шаблонов или создания новых. Таким образом, все поставленные задачи в ходе работы были выполнены, а цель работы можно считать достигнутой.

Библиографический список

1. Большакова Е.И., Клышинский Э.С., Ландэ Д.В., Носков А.А., Пескова О.В., Ягунова Е.В., Автоматическая обработка текстов на естественном языке и компьютерная лингвистика, М.: МИЭМ, 2011. — 272 с.
2. Крештель Е.В. Алгоритмы выделения ключевых слов в текстах на естественных языках /Е. В. Крештель, А. А. Кретов., И. Е. Воронина //Информатика: проблемы, методология, технологии :матер. 3-й регион. науч.-метод. конфер., Воронеж, 2001 г. — Воронеж. : Изд-во ВГУ, 2001 . — С. 35.
3. Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 63-70.
4. Документация Stanford CoreNLP: http://stanfordnlp.github.io/CoreNLP/ (дата последнего обращения 15.05.2016)

5. Документация The Stanford Parser: http://nlp.stanford.edu/software/lex-parser.html (дата последнего обращения 06.06.2016)
6. D. Jurafsky, James H. Martin. Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition. Prentice-Hall, 2000.
7. C. Manning, H. Schutze. Foundations of Statistical Language processing. The MIT Press, 1999.

Приложение 1. Программный код реализации класса HypGenerator

```
public class HypGenerator {

    final CoreNlpPipeline mCoreNlpPipeline;
    BaseHypothesisRule.CoreNlpRulesCallback
mCoreNlpRulesCallback;
    BaseHypothesisRule rulesList[] = new
BaseHypothesisRule[9];
    NumberProcessingRule mNumberProcessingRule;
    public PunctuationRule mPunctuationRule;
    SProcessingRule mSProcessingRule;

    public HypGenerator() {

        mCoreNlpPipeline = new CoreNlpPipeline();
        mCoreNlpRulesCallback = new
BaseHypothesisRule.CoreNlpRulesCallback() {
            @Override
            public Tree getNewTree(Tree oldTree) {
                return
mCoreNlpPipeline.getTree(CoreNlpOutput.getSentenceFromTree(ol
dTree));
            }

            @Override
            public Tree getNewTree(String sentence) {
                return mCoreNlpPipeline.getTree(sentence);
            }
        };

        rulesList[0] = new JJbeforeNounRule();
        rulesList[1] = new DatePeriodRule();
        rulesList[2] = new NumeralRule();
        rulesList[3] = new AdverbRule();
        rulesList[4] = new ProperNounRule();
        rulesList[5] = new SimilarLeavesSimpleRule();
        rulesList[6] = new INprocessingRule();
        rulesList[7] = new INinsideINRule();
        rulesList[8] = new INremovingRule();
```

```
        for (int i = 0; i < rulesList.length; i++ ){

rulesList[i].setCoreNlpRulesCallback(mCoreNlpRulesCallback);
        }

        mNumberProcessingRule = new NumberProcessingRule();

mNumberProcessingRule.setCoreNlpRulesCallback(mCoreNlpRulesCa
llback);
        mPunctuationRule = new PunctuationRule();

mPunctuationRule.setCoreNlpRulesCallback(mCoreNlpRulesCallbac
k);
        mSProcessingRule = new SProcessingRule();

mSProcessingRule.setCoreNlpRulesCallback(mCoreNlpRulesCallbac
k);

    }

    public List<InputHypothesis> generateHypothesis(String
inputSentence) {

        List<InputHypothesis> inputHypothesises = new
ArrayList<InputHypothesis>();
        List<InputHypothesis> results = new
ArrayList<InputHypothesis>();

        List<Tree> sentencesTree =
mCoreNlpPipeline.getTrees(inputSentence);

        for (Tree sentence : sentencesTree) {
            inputHypothesises.add(new
InputHypothesis(sentence, 1.0));
        }

        inputHypothesises =
mNumberProcessingRule.getHypothesis(inputHypothesises);
        inputHypothesises =
mSProcessingRule.getHypothesis(inputHypothesises);

        for (int i = 0; i < inputHypothesises.size(); i++) {

            InputHypothesis hyp = new InputHypothesis();

hyp.setHTree(mCoreNlpPipeline.getTree(mPunctuationRule.remove
```

```
Punctuation(CoreNlpOutput.getSentenceFromTree(inputHypothesis
es.get(i).getHTree())));

hyp.setHConfidence(inputHypothesises.get(i).getHConfidence())
;

            results.add(hyp);
        }

        for (int i = 0; i < rulesList.length; i++) {
            results = rulesList[i].getHypothesis(results);
        }

        Collections.sort(results);

        return results;
    }

}
```

Приложение 2. Программный код реализации класса InputHypothesis

```
public class InputHypothesis implements
Comparable<InputHypothesis>{

    private Tree hTree;
    private HypothesisConfidence hConfidence;

    public InputHypothesis() {
        hTree = null;
        hConfidence = new HypothesisConfidence();
    }

    public InputHypothesis(Tree tree, double confidence) {
        hTree = tree;
        int wCount = tree.getLeaves().size();
        int tDeep = tree.depth();
        hConfidence = new HypothesisConfidence(wCount, tDeep,
confidence);
    }

    public InputHypothesis(Tree tree, HypothesisConfidence
hypothesisConfidence) {
        hTree = tree;
        hConfidence = hypothesisConfidence;
        hConfidence.setWordCount(tree.getLeaves().size());
        hConfidence.setTreeDeep(tree.depth());
    }

    public Tree getHTree() {
        return hTree;
    }
    public void setHTree(Tree tree) {
        hTree = tree;
    }

    public HypothesisConfidence getHConfidence() {
        return hConfidence;
    }
    public void setHConfidence(HypothesisConfidence
confidence) {
        hConfidence = confidence;
```

```
    }

    @Override
    public int compareTo(InputHypothesis o) {
        if (this.getHConfidence().getConfidence() >
o.getHConfidence().getConfidence()) {
            return -1;
        }
        if (this.getHConfidence().getConfidence() <
o.getHConfidence().getConfidence()) {
            return 1;
        }
        return 0;
    }
}
```

Приложение 3. Программный код реализации класса HypothesisConfidence

```
public class HypothesisConfidence {
    private int wordCount;
    private int treeDeep;
    private double confidence;

    public HypothesisConfidence(){
        wordCount = 0;
        treeDeep = 0;
        confidence = 0.0;
    }

    public HypothesisConfidence(int wCont, int tDeep, double
conf){
        wordCount = wCont;
        treeDeep = tDeep;
        confidence = conf;
    }

    public int getWordCount() {
        return wordCount;
    }

    public void setWordCount(int wordCount) {
        this.wordCount = wordCount;
    }

    public int getTreeDeep() {
        return treeDeep;
    }

    public void setTreeDeep(int treeDeep) {
        this.treeDeep = treeDeep;
    }

    public double getConfidence() {
        return confidence;
    }

    public void setConfidence(double confidence) {
        this.confidence = confidence;
```

```
    }

    public HypothesisConfidence copy(){
        return new HypothesisConfidence(wordCount, treeDeep,
confidence);
    }

    public void updateConfidence(int rwc, double POSc, double
RuleCoeff) {
        double confVal = getConfidence();
        confVal -= confVal
                *((double)(rwc)/(double)getWordCount())
                * POSc
                * RuleCoeff;

        if (confVal < 0 ) {
            confVal = 0.0;
        }
        setConfidence(confVal);
    }

    public void updateConfidence(int rwc, String POSname,
double RuleCoeff) {
        updateConfidence(rwc,
CoreNlpConstants.getPOScoefficient(POSname), RuleCoeff);
    }

    public void updateConfidence(int rwc, double RuleCoeff) {
        updateConfidence(rwc, 1.0, RuleCoeff);
    }

    public void updateConfidence(int rwc, String POSname) {
        updateConfidence(rwc,
CoreNlpConstants.getPOScoefficient(POSname), 1.0);
    }
}
```