МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

РЕКОМЕНДОВАНО К ЗАЩИТЕ
В ГЭК И ПРОВЕРЕНО НА ОБЪЕМ
ЗАИМСТВОВАНИЯ
Заведующий кафедрой
к.т.н., доцент
_____М. С. Воробьева
24.06 2019 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(магистерская диссертация)

РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ КОНСУЛЬТИРОВАНИЯ ДЛЯ ЦЕНТРОВ ГОСУСЛУГ «МОИ ДОКУМЕНТЫ» ТЮМЕНСКОЙ ОБЛАСТИ

02.04.03. Математическое обеспечение и администрирование информационных систем

Магистерская программа «Разработка, администрирование и защита вычислительных систем»

Выполнил работу
Студент 2 курса
очной формы обучения
Овчаренко
Денис
Игоревич

Руководитель работы
к.п.н., доцент кафедры
Программного обеспечения
Плотоненко
Юрий
Анатольевич

Рецензент
Заместитель директора
ГАУ ТО "МФЦ"
Сальников
Дмитрий
Евгеньевич

Тюмень 2019

Содержание

Введение

Консультирование заявителей по вопросам оказания услуг является неотъемлемой частью работы филиалов центров госуслуг «Мои Документы». В связи с этим, специалистам приходится ежедневно обрабатывать большие объемы данных, тратя человеко-часы, чтобы предоставить ответы заявителям. Важным фактором также является то, что вопросы имеют общую предметную область и зачастую повторяются, имея аналогичный смысл, но отличный по формулировке запрос, заставляя специалиста повторно составлять на них ответы. Таким образом, особую актуальность приобретает задача разработки интеллектуальной, автоматизированной, вопросно-ответной системы, позволяющей анализировать входящие вопросы и, опираясь на собранную базу знаний, предоставлять на них ответы. Разработка вопросно-ответных систем ведётся с 60-х годов (были разработаны системы Baseball, Lunar) и по сей день (Siri, WolphramAlpha, Алиса) [1-5]. Вопросы создания таких систем рассматривались в работах отечественных специалистов и исследователей (Науменко А. М., Шелудько С. Д., Юлдашев Р. Ю., Хлебников Н. О, Лапшин В. А., Мочалова А. В., Мочалов В. А., Медовый В. С., Бухаров М.Н., Кулешов А.С., Беляев С. А. и др.) [6-12]. Целью выпускной квалификационной работы являются разработка интеллектуальной системы консультирования пользователей центра госуслуг «Мои Документы»

Поставленная цель определила следующие задачи:

- Исследовать предметную область.
- Провести анализ алгоритмов работы вопросно-ответных систем и изучить необходимые инструменты для их реализации.
- Выделить основные компоненты проектируемой системы и определить архитектуру их взаимодействия, с последующей реализацией.

1. Анализ вопросно-ответных систем и инструментов обработки естественного языка

1.1 Определение вопросно-ответной системы

Ежегодно, дистанционное консультирование заявителей по вопросам оказания услуг становится все более востребованным в центрах госуслуг «Мои Документы». Количество обращений с помощью телефонных звонков и электронных сообщений растет, тем самым, увеличивая нагрузку на специалистов. Основной проблемой является то, что вопросы зачастую схожи по смыслу, но различаются по формулировке. Данная проблема создает дополнительную нагрузку на специалистов, так как они должны каждый раз заново отвечать на одни и те же вопросы. Внедрение вопросно-ответной системы может решить данную проблему.

В своей статье В.А Лапшин дал следующее определение вопросно-ответной системе – программный модуль, позволяющий человеку вести с машиной диалог на естественном языке [7]. Представлено это в том виде, что пользователь задает вопросы программной системе, а программная система предоставляет ответы, формируемые в виде осмысленных предложений.

Похожее определение дали А.С. Кулешов и С. А. Беляев [11]. По их мнению, вопросно-ответная система является комплексом интеллектуальных и справочных систем, использующих естественно языковой интерфейс.

Вопросно-ответные системы принято разделять на два класса:

- Узкоспециализированные – это системы, работающие в конкретных областях ограничиваясь определенным источником знаний. Примерами реализации являются Baseball, WolframAlpha [1,4].
- Общие - работают с информацией по всем областям знаний, что в свою очередь позволяет вести поиск в смежных областях. Примерами реализации являются Siri, Алиса [3,5].

В свою же очередь, К. В. Ненаусников, С. В. Кулешов и А. А. Зайцева классифицировали вопросно-ответные системы по способу реализации:

- на основе логического вывода по онтологиям,
- правилам и основам синтаксиса,
- использование искусственных нейронных сетей
- на основе показателя удовлетворённости пользователя [12].

Соловьёв А.А и Пескова О.В в своей работе также выделяют четыре класса систем:

- Метапоисковая система – архитектура такой системы предусматривает использование существующей классической поисковой системы в качестве источника данных.
- Система поиска по аннотированному тексту – имеют в своём составе поисковый индекс документов, который дополняется определенными атрибутами. Элементами данного индекса являются не отдельные слова заданного вопроса, а объекты детального лингвистического анализа.
- Экспертная система - построена по принципу работы со структурированными базами данных;
- Система поиска в коллекциях вопросов и ответов – система, позволяющая пользователям, имеющий информационную потребность, открывать страницу веб-сайта системы и формулировать вопрос. Система ищет похожие вопросы в коллекции вопросов и ответов и выдаёт найденный раздел, где обсуждается вопрос [13].

Также, рядом работ было отмечено, что современные вопросно-ответные системы содержат особый модуль — классификатор вопросов [6,7,8,11]. Задачей данного модуля является определение типов вопроса и ожидаемого на него ответа.

1.2 Исследования открытых вопросно-ответных систем

В 2002 году группа исследователей провела изучение вопросно-ответных систем. В ходе исследований были отмечены некоторые особенности, которые необходимо учитывать во время реализации вопросно-ответной системы [14].

- Типизация вопросов – разные вопросы требуют разные способы поиска ответов. Перед реализацией вопросно-ответной системы нужно заранее определить типы возможных вопросов.
- Обработка вопросов – нередки случаи, когда вопросы схожи по смыслу, но различаются по формулировке. В связи с этим возникают ситуации, когда одну и ту же информацию можно запросить несколькими способами. Решением данной проблемы может быть реализация функционала, с помощью которого система сможет распознавать равнозначные по смысловой нагрузке вопросы, независимо от используемых синтаксических взаимосвязей, слов, и идиом. Также необходимо предусмотреть возможность декомпозировать сложные вопросы на простые.
- Определение контекста вопросов – если удастся определить контекст запроса, то можно устранить двусмысленность заданного вопроса.
- Создание базы знаний для вопросно-ответной системы – перед реализацией вопросно-ответной системы необходимо подготовить актуальные источники знаний. Без данных источников невозможно предоставить правильный ответ, какие бы способы обработки текстов ни применялись.
- Выделение и формулировка ответа – ответ, предоставленный системой, должен выглядеть естественно. В частных случаях достаточно просто выделить ответ из текста. К примеру, если требуется наименование, величина или дата – то достаточно предоставление прямого ответа. Но иногда приходится иметь дело со сложными запросами, и здесь необходима реализация алгоритмов объединения ответов из разных источников.

- Ответы в реальном времени – система должна оперативно предоставлять ответы, независимо от сложности и двусмысленности вопроса, а также размера базы знаний.
- Поддержка многоязыковых запросов - система должна уметь переводить вопросы и предоставляемые ответы.
- Интерактивность системы - часто ответы, предлагаемые вопросно-ответной системой, предоставляют неполную информацию. Возможно, система неправильно определила тип вопроса. В этом случае пользователю необходимо предоставить возможность переформулировать свой запрос.
- Ввод профилей пользователей - системе могут понадобиться сведения о пользователе, его характеристиках, которые могли бы увеличить производительность системы.
- Механизм рассуждений - система не должна ограничиваться только теми ответами, которые хранятся в доступных ей источниках знаний. Для этого в вопросно-ответной системе необходимо реализовать функцию обучения, с помощью которой система сможет получать новые знания.

1.3 Классификация вопросов

Одним из важнейших компонентов вопросно-ответной системы является модуль классификации вопроса. Его задача состоит в том, чтобы соотнести вопрос с одним из предопределенных классов. Связь между вопросами и их классами позволяет определить источник информации, в котором необходимо искать ответ на данный вопрос, а также выбрать алгоритм, с помощью которого этот поиск будет производиться. Множество категорий, используемых для классификации вопросов, можно назвать онтологией [16]. Онтология вопросов обычно представляет собой иерархическую структуру – таксономию, понятия верхнего уровня которой имеют довольно специфический вид, основанный на функциональном предназначении вопроса.

Задача выделения классов запросов впервые была затронута в работе В. Ленерт, вышедшей во второй половине 70-х годов прошлого века [17]. В. Ленерт представила публике достаточно подробное исследование, в котором была разработана модель представления семантики текстовых выражений на основе теории так называемых «понятийных зависимостей» (Conceptual Dependency).

В. Ленерт предложила 13 категорий, в которые попадают все возможные вопросы, которые может задать пользователь. Эти категории, вместе с примерами вопросов, которые им принадлежат, приведены в таблице 1.

Таблица 1

Категории вопросов В. Ленерт

Категория вопроса	Примеры вопросов
Причина	Почему Олег приехал в Москву?
Цель	Для чего Олег взял эту книгу?
Возможность	Что должен сделать Олег, чтобы уехать?
Контроль	Олег уехал?
Дизъюнкция	Были ли здесь Василий или Олег?
Процедура	Каким образом Олег добрался до Москвы?
Дополнение	Что ел Олег?
Ожидание	Почему Олег не приехал в Москву?
Суждение	Что должен сделать Василий, чтобы Олег не уехал?
Квантификация	Сколько людей собралось на этом стадионе?
Свойство	Какого цвета глаза у Олега?
Запрос	Не передадите мне соль?

Система классов, введенная В. Ленерт, была расширена А. Гессером в его работе [18]. К существующим 13 категориям были добавлены еще пять: Определение, Пример, Интерпретация, Отсутствие и Сравнение.

Приведенная классификация является довольно общей и пригодна только для первичной обработки вопроса, например, для выбора алгоритма обработки запроса.

1.4 Классификация в машинном обучении

Машинное обучение — это раздел искусственного интеллекта, цель которого является изучающий методы построения алгоритмов, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач. Для построения таких алгоритмов используются средства математической статистики, численных методов, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме.

В машинном обучении задача классификации решается, в частности, с помощью методов искусственных нейронных сетей при постановке эксперимента в виде обучения с учителем. Данный подход подразумевает, что имеется конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется обучающей выборкой. Классовая принадлежность остальных объектов не известна, и для ее определения необходимо построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Наивная байесовская классификация

Наивный байесовский классификатор – это алгоритм классификации, основанный на теореме Байеса с допущением о независимости признаков. Теорема Байеса позволяет рассчитать апостериорную вероятность $P(c|x)$ на основе $P(c)$, $P(x)$ и $P(x|c)$:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \quad (1)$$

где

P(c) – априорная вероятность гипотезы c;

P(c|x) – вероятность гипотезы c при наступлении события x;

P(x|c) – вероятность наступления события x при истинности гипотезы A;

P(x) – полная вероятность наступления события x.

В зависимости от точной природы вероятностной модели, наивные байесовские классификаторы поддаются эффективному обучению. Во многих практических приложениях для оценки параметров для наивных байесовых моделей используют метод максимального правдоподобия.

Модели на основе наивного байесовского алгоритма достаточно просты и крайне полезны при работе с очень большими наборами данных. При своей простоте алгоритм способен превзойти даже некоторые сложные алгоритмы классификации.

Если представить модель в контексте вопросно-ответной системы, то уравнение модели будет иметь следующий вид:

$$c = \arg max_c P(O|C) \quad (2)$$

где

O – вопрос в текстовом виде;

C – набор классов, каждый которого включает в себя список инверсий определённого вопроса и интересующий ответ на него;

c – класс, имеющий максимальную вероятность соответствия к заданному вопросу.

Так как вычислить уравнение *P(C|O)* при большом количестве свойств очень сложно, обратившись к теореме Байеса, можно представить его в следующем виде:

$$P(C|O) = \frac{P(C)P(O|C)}{P(O)} \quad (3)$$

Раз необходим максимум от функции, то на практике интересен лишь числитель (знаменатель является константой). Также, текстовую строку вопроса *O* можно представить в виде набора признаков *o*, таким образом уравнение принимает следующий вид:

$$P(C|o_1, \dots, o_n) = \frac{P(C)P(o_1, \dots, o_n|C)}{P(o_1, \dots, o_n)} \quad (4)$$

где числитель принимает вид:

$$P(C)P(o_1|C)P(o_2|Co_1) \dots P(o_2|Co_1o_2 \dots o_2) = P(C)\prod_i(o_i|C) \quad (5)$$

Следовательно, формула модели выглядит следующим образом:

$$c = \arg max_{c \in C} P(c|o_1o_2 \dots o_2) = \arg max_{c \in C} P(c)\prod_i P(o_i|c) \quad (6)$$

Для нахождения класса c необходимо вычислить параметры $P(C)$ и $P(O|C)$. Процедура вычисления этих параметров называется обучение классификатора.

Положительными сторонами данного алгоритма являются быстрота классификации и небольшое количество обучающей выборки для ее корректной работы.

Слабой стороной алгоритма является тот факт, что если в тестовой выборке присутствует некоторое значение признака категорий, которое не встречалось в обучающей выборке, то модель присвоит нулевую вероятность значения и не сможет дать точный прогноз. Это явление известно, как «нулевая частота».

Решением дайной проблемы может стать применение адаптивного сглаживания (сглаживание Лапласа).

Дерево принятия решении

Дерево принятия решений – модель, представляющая собой совокупность правил для принятия решений. Идея данной классификации состоит в том, чтобы осуществлять процесс деления исходных данных на группы, пока не будут получены однородные множества.

Структура дерева представляет собой узлы, листья и ветки. В узлах происходит ветвление процесса в зависимости от сделанного выбора. Конечные узлы называются листьями. Листья являются конечными результатами последовательного принятия решений. Чтобы классифицировать новый случай, необходимо пройтись по дереву до листа и вернуть соответствующее значение. Цель данной модели состоит в том, чтобы предсказывать значения целевой переменной на основе нескольких переменных, полученных на входе. Классификация на основе данного алгоритма используется в таких задачах, как распознавание текста, информационного поиска, распознавания речи, анализе изображений, распознавание жестов и прочих похожих задачах [23].

На *рисунке 1* схематично представлены элементы и процесс простого, бинарного алгоритма. Кругом обозначены узлы, прямоугольником – листы.

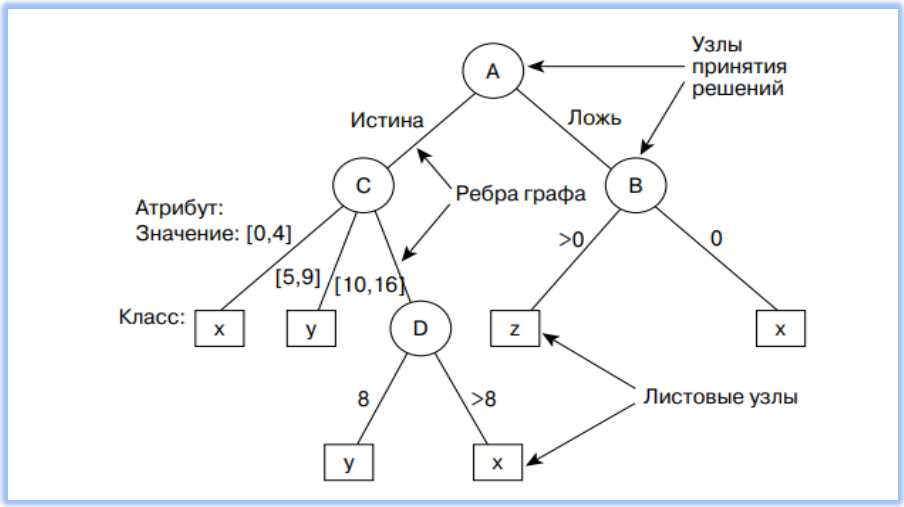

Рис 1. - Метод дерево принятие решений

Метод дерево принятие решений реализует принцип рекурсивного деления (или принцип «Разделяй и властвуй»). В узлах, начиная с корневого, выбираются признаки, значения которых используются для разбиения всех данных на 2 класса. Процесс повторяется до получения критерия остановки:

1) Все данные узла принадлежат одному и тому же классу;
2) Невозможно дальнейшее разбиение из-за отсутствия признаков;
3) Дерево превысило ранее заданный уровень вложенности.

Модель алгоритма может быть представлена следующим образом:

$$(x, Y) = (x_1, x_2, .. x_n, Y) \quad (7)$$

где

Y – вопрос, который необходимо классифицировать;

x – вектор, состоящий из переменных x_1, x_2 и т.д. которые используются для выполнения классификации.

Алгоритм обладает следующими достоинствами:

- простая интерпретация,
- не требует предварительной обработки данных,
- хорошая производительность,
- возможность работы как с категориями, так и с количественными значениями;

К недостаткам можно отнести:

- Нестабильность процесса. Нередко небольшие изменения в наборе данных могут приводить к построению совершенно другого дерева. Это связано с иерархичностью дерева. Изменения в узле на верхнем уровне ведут к изменениям во всем дереве ниже;
- сложность контроля размера дерева. Размер дерева является критическим фактором, определяющим качество решения задачи;

- неадекватность разделения на классы в сложных случаях.

Метод опорных векторов

Основная идея метода заключается в построении гиперплоскости, разделяющей объекты выборки наиболее оптимальным способом. Алгоритм работает в предположении, что чем больше расстояние (зазор) между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет средняя ошибка классификатора. Данный метод позволяет решать задачи бинарной классификации. Сам метод относится к семейству линейных классификаторов.

Идея метода опорных векторов заключается в том, что объекты обрабатываемых данных представляются как векторы (точки) в p-мерном пространстве. Каждая из этих точек принадлежит к одному из двух классов. Для разделения классов строят гиперплоскость размерности (p-1). Искомых гиперплоскостей может быть много, алгоритм находит оптимальную, которая максимизирует зазор между классами, что способствует более уверенной классификации [24].

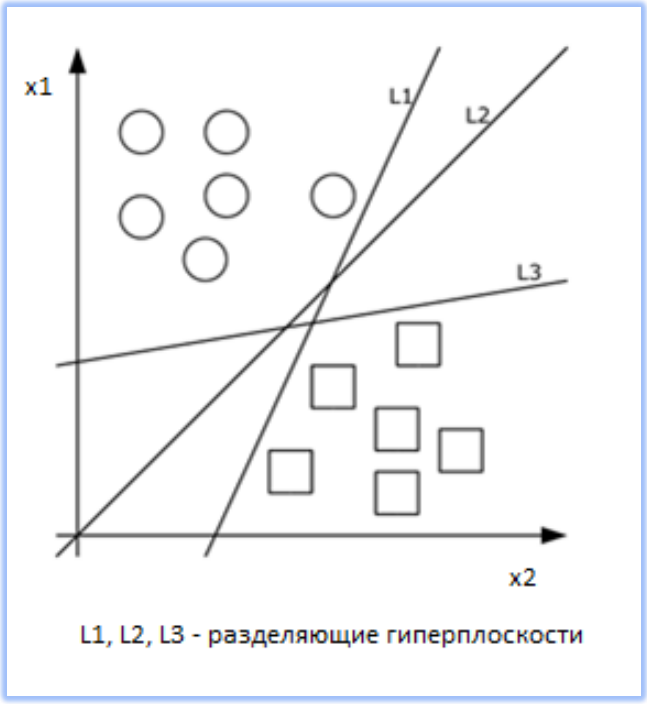

Рис. 2 – Разделение классов

На рисунке 2 представлено несколько классифицирующих разделяющих прямых (гиперплоскостей), из которых только одна (L2) соответствует оптимальному разделению.

Модель данного алгоритма может быть представлена следующим образом:

$$a(x) = sign\left(\Sigma_{j=1}^{n} w_j x_j - w_0\right) = sign(\langle w, x\rangle - w_0) \quad (8)$$

где

x = $(x_1, \dots, x_n)$ – признаковое описание объекта x;

w – вектор;

w_0 – параметры алгоритма;

$\langle w, x\rangle$ –разделяющая гиперплоскость в пространстве R^n.

Процесс построения разделяющей гиперплоскости является задачей квадратичного программирования и в конечном итоге сводится к поиску седловой точки функции Лагранжа, что, в свою очередь, приводит к задаче нелинейной оптимизации с ограничениями:

$$\begin{cases} -L(\lambda) = -\sum_{i=1}^{l} \lambda_i + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \lambda_i \lambda_j y_i y_j \langle x_i, x_j\rangle \to min_\lambda; \\ 0 \le \lambda_i \le C, i = 1, \dots, l; \\ \sum_{i=1}^{l} \lambda_i y_i = 0. \end{cases} \quad (9)$$

К достоинствам данного алгоритма можно отнести высокое быстродействие. Также метод находит разделяющую полосу максимальной ширины, что позволяет в дальнейшем осуществлять более уверенную классификацию.

К недостаткам можно отнести низкую выразительность метода и необходимость предварительно обработки данных.

Метод максимизации энтропии

Энтропия — это мера априорной неопределенности системы. Энтропия имеет ряд полезных свойств:

- обращение в ноль при условии, что одно из состояний системы достоверно, а другие – невозможны;
- обращение в максимум, если состояния, при их заданном числе, равновероятны;
- суммирование при объединение их независимых систем.

Принцип максимизации энтропии утверждает, что если плотность распределения некоторой случайной величины неизвестна, то из логических соображений следует выбрать такую плотность распределения, которая обеспечивает максимизацию энтропии случайной величины при учете всех известных ограничений. Применение этого критерия приводит к решению, отличающемуся минимальным смещением, так как плотность распределения любого другого вида будет обладать большим смещением в сторону информации, содержащейся в известном наборе данных.

Согласно принципу максимизации энтропии, вид модели $q(x)$ подбирается таким образом, чтобы максимизировать предмет энтропии $H(q)$, не делая никаких дополнительных предположений для последовательности из N слов, не представленных в обучающей выборке. Принцип максимизации энтропии можно представить в следующем виде:

$$H(q) = -\Sigma_x q(x) log q(X) \quad (10)$$

Данный алгоритм используется для решения широкого спектра задач классификации текста, таких как определение языка, классификация тем, эмоциональной окраски и т.д. Также, метод максимизации энтропии был выбран для построения классификации в работе [25].

Кроме рассмотренных алгоритмов существую другие: метод N-грамм, классификация на основе персептрона, логистическая регрессия, лес решений и т.д. Данный тип классификации часто используется во время обработки (анализа) естественного языка.

1.5 Обработка естественного языка

На данный момент, современные вопросно-ответные системы базируются на принципах NLP (Natural Language Processing), с помощью которых они реализуют совокупность классов интеллектуальных систем информационного поиска. Системы данного типа в основном имеют общие принципы обработки естественного языка. Но есть и различия, а именно разные способы внутреннего представления семантического содержимого, а также различные алгоритмы и методы для преобразования текста на естественном языке к внутреннему представлению системы. И от выбора алгоритма компьютерной лингвистик напрямую зависит результат работы вопросно-ответной системы.

Помимо классификации, обработка естественного языка включает в себя следующие этапы:

- перевод всех символов текста в единый регистр;
- удаление лишних пробелов, знаков пунктуации цифр (или их преобразование в текстовый эквивалент);
- удаление стоп слов;

- токенизация;
- нормализация**;**
- стемминг;
- лемматизация;
- векторизация.

Токенизация

Токенизация — это процедура разбиения текста на более мелкие части - токены. Токенами могут быть слова и знаки пунктуации. Основная задача токенизация - представить текст в виде массива значимых слов. После токенизации, как правило производится фильтрация, в ходе которой удаляются и не значимые слова, в основном предлоги.

Нормализация

Процесс нормализации, реализованный в грамматическом словаре, позволяет убрать из исходного текста грамматическую информацию (падежи, числа, глагольные виды и времена, залоги причастий, род и так далее), оставляя смысловую составляющую. Два других схожих процесса - стемминг и лемматизация - пытаются достичь такого же эффекта, но глубина преобразования текста в них меньше.

Стемминг

Стемминг — это нахождение основы слова – стеммы. Стемма - часть слова, которая передает его лексическое значение. Стемминг позволяет привести слово к его основной форме. Идея данного подхода состоит в нахождении основы слова, для этого отбрасываются окончания и суффиксы. В основном, стеминг базируется на правилах морфологии языка и не требует создания дополнительных словарей. Правила работы стеммера создаются заранее, часто для решения данной задачи используются регулярные выражения. Более сложным методом решения проблемы определения основы слова является лемматизация.

Лемматизация

Процесс, являющийся альтернативой стемминга. Задача лемматизации - преобразование слова в словарный вид, лемму. Для русского языка процесс лемматизациий заключается в том, что существительные и прилагательные приводятся к форме именительного падежа единственного числа, а глаголы, причастия и деепричастия преобразуются в глаголы неопределенной формы.

Векторизация

Большинство математических моделей работают в векторных пространствах больших размерностей, поэтому необходимо преобразовывать текст в такой вид, который можно отобразить в векторном пространстве. Одним из наиболее простых методов преобразования текстов на естественном языке является метод «мешок слов». Основной объект данной модели является слово. Оно содержит единственный атрибут - частоту встречаемости этого слова в тексте. В модели текста «мешок слов» учитывается только количество вхождений конкретных слов в исходном тексте, при этом игнорируются: порядок слов в документе, морфологические формы представления слов. Наиболее распространенным методом для вычисления признака является мет од TF-IDF, модификация метода «мешок слов». В основу данного метода положено предположение, что слова в документе имеют разную значимость, и если слово встречается в небольшом числе документов, то оно является важным для них. TF вычисляется, как доля документов, в которых присутствует токен, а IDF как инверсия частоты, с которой некоторое слово встречается в документах коллекции. Вес слова в документе вычисляется как произведение TF и IDF.

Существует ряд NLP-инструментов, таких как StanfordNLP, OpenNLP, Natural Language Toolkit (NLTK) и т.п.

Stanford CoreNLP

Библиотека предоставляет инструменты для доступной обработки естественного языка, которые могут быть интегрированы в приложения, в зависимости от нужд пользователя. Она может дать начальные формы слов, их части речи, например, имена компаний, людей и т.д., разметить структуру предложений, указать зависимые слова, показать какие слова относятся к одной и той же сущности, показать настроения пользователя (для анализа отзывов). Инструменты библиотеки поддерживают английский и китайский языки. CoreNlp широко используются в промышленности, академиях и правительствах. Библиотека активно разрабатывается и наши дни. Инструменты Standfort выпускаются на условиях лицензии GPL, но с запретом использования в коммерческих приложениях.

LingPipe

Библиотека LingPipe состоит из набора инструментов для выполнения общих задач обработки естественного языка и поддерживает обучение и тестирование моделей. Существуют две версии LingPipe – свободная и коммерческая. Использование свободной версии подразумевает определенные ограничения [21].

GATE

General Architecture for Text Engeneering (GATE) – это набор инструментов на Java, разработанный в университете города Шеффилд, поддерживающий многие языки и типы задач NLP. GATE также можно использовать в качестве конвейера для NLP – обработки. Пакет инструментов GATE включает в себя следующие компоненты:

- Gate Developer – среда разработки приложений для обработки текстов, включает в себя встроенную систему извлечения информации;

- Gate Cloud – набор инструментов для облачных вычислений;
- Gate Teamware – веб-приложение, позволяющие проводить совместную разработку проектов;
- Gate Mimir – мультипарадигмальный инструмент для поиска и индексирования данных;
- Gate Embedded – вспомогательный фреймворк, предоставляющий доступ к инструментам Gate Developer [21].

GATE постоянно поддерживается обширным сообществом, включающим в себя разработчиков, пользователей, преподавателей, студентов и учёных. Данная библиотека используется как в коммерческих, так и в научно-исследовательских проектах крупными корпорациями, научно-исследовательскими лабораториями и университетами, коммерческими предприятиями малого и среднего бизнеса по всему миру. Применение GATE для решение многих задач обработки естественного языка требует написания небольшого объема исходного кодах [22].

OpenNLP

Библиотека Apache OpenNLP представляет собой инструмент машинного обучения, предназначенный для обработки текста на естественном языке. Она поддерживает наиболее распространённые задачи NLP, такие как токенизация, сегментирование предложений, пометка частей речи, извлечение именованных сущностей, анализ и разрешение кореферентности. Эти задачи обычно используются для создания сложных служб обработки текста. Цель проекта OpenNLP заключается в создании зрелого инструментария для решения вышеупомянутых задач. Дополнительной целью является предоставление большого числа готовых моделей для различных языков, а также аннотированных текстовых ресурсов, из которых производятся эти модели. Библиотека Apache OpenNLP содержит несколько компонентов, позволяющих построить полный

конвейер обработки естественного языка. Так же эти компоненты можно обучить и оценить модель. Каждое средство доступно через интерфейс прикладных программ. Кроме того, для удобства экспериментов и обучения, доступен интерфейс командной строки (CLI). Пакет Apache OpenNLP распространяется на условиях лицензии Apache License. Исходные коды доступны на официальном сайте проекта. Существует достаточно полная документация как для пользователя, так и для разработчика интегрированных решений [20].

NLTK

Библиотека Natural Language Toolkit (NLTK) – набор инструментов и программ для символьной и статистической обработки естественного языка, написанных на языке программирования Python. Содержит графические представления и примеры данных. Сопровождается обширной документацией, включая книгу с объяснением основных концепций, стоящих за теми задачами обработки естественного языка, которые можно выполнять с помощью данного пакета. NLTK подходит для задач, подразумевающих изучение компьютерной и эмпирической лингвистик, когнитивистики, искусственного интеллекта, информационного поиска и машинного обучения. NLTK с успехом используется в качестве учебного пособия, в качестве инструмента индивидуального обучения и в качестве платформы для прототипирования и создания научно-исследовательских систем. Предоставляется NLTK как свободное программное обеспечение.

Сравнение перечисленных инструментов представлены в Приложение А.

1.6 Архитектура вопросно-ответной системы

В своей работе «Семантический анализатор русскоязычного текста для вопросно-ответной системы» Мочаловой А.В. обобщила принципы работы исследованных вопросно-ответных систем [15].

На вход системе подается текст вопроса, сформулированный на естественном языке. Затем текст вопроса проходит автоматическую обработку, основными этапами которой являются:

- предварительная обработка текста, включающая удаление лишних символов форматирования, исправление орфографических и пунктуационных ошибок, перевод всех букв в тексте в единый регистр (нижний или верхний), удаление лишних пробелов и символов переноса строк и т.п.;
- извлечение именованных сущностей, разбиение текста на предложения, токенизация (разбиение предложений на слова);
- морфологический, синтаксический и семантический анализ, включающий в себя процессы лемматизации или стемминга, а также векторизации текста, в случае ее необходимости.

Следящим шагом является анализ текста заданного вопроса. Часто, для решения данной задачи используются различные структурированные и лингвистические ресурсы: словари, базы знаний, базы фактов, онтологии.

Затем текст вопроса классифицируется в соответствии с принятым в данной системе методом классификации. Основываясь на результатах обработки, анализа и классификации вопроса формируется запрос модулю генерации ответа.

Модуль генерации ответов выбирает определенное количество документов (если поиск производится не по одному заданному документу), наиболее подходящие запросу. Поиск документов может производиться с помощью внешних поисковых систем, либо с помощью собственных поисковых инструментов, являющимися частью разрабатываемой системы. Найденные документы также, как и вопросы, проходит автоматическую обработку текста. При этом алгоритмы обработки текста вопроса могут отличаться от алгоритмов автоматической обработки набора документов, выбранных поисковым модулем системы.

Далее, посредством внутренних алгоритмов работы вопросно-ответной системы, происходит поиск конкретных фрагментов текстов из документов,

переданных поисковой системой. Выбранные фрагменты текста представляются системой в качестве ответа. Наиболее развитые вопросно-ответные системы на этапе выбора фрагментов текста могут использовать данные из структурированных лингвистических ресурсов.

В своей работе Мочалова А.В. подчеркнула особую важность модуля обработки текста, так как от него зависит корректность работы всей системы.

Схожая схема архитектуры предложена в работе Науменко А.М, Шелудько С. Д., Юлдашева Р.Ю, Хлебникова Н.О и Радыгина В.Ю. [6]. Авторы работы предлагают заменить модуль генератора ответа на использование базы часто задаваемых вопросов в расширенном виде, где вопросы соответствуют уже готовым ответам. Таким образом, описанный подход сводит задачу поиска ответов к поиску похожего вопроса из базы, и предоставления соответствующего готового ответа.

2. Разработка вопросно-ответной системы

2.1 Требования к системе

Предложенная для разработки вопросно-ответная система должна удовлетворять следующим условиям:

- Наличие интерфейсов для интеграции других ПО.
- Определение тематики задаваемых вопросов и предоставление ответов на них.
- Перенаправление вопросов операторам в случае, если система не смогла их распознать.
- Механизм самообучения после предоставления удовлетворительного ответа.
- Возможность вести диалог в рамках одного вопроса, с помощью которого возможно задавать дополнительные, уточняющие вопросы.

Для разработки данной системы необходимо было выбрать и реализовать следующие компоненты:

- Хранилище данных для обучающей и тестовой выборок, а также прочей информации, необходимой для корректной работы разрабатываемой системы.
- Инструменты для обработки текста, с помощью которых можно будет проанализировать и классифицировать поступающие вопросы.
- Модуль обмена информацией между компонентами системы (веб-сервисы, брокер сообщений).
- Программное обеспечение, связывающее все остальные компоненты системы.

2.2 Выбор технологий и разработка архитектуры системы

Для решения задачи создания интеллектуальной системы консультирования было решено реализовать систему с клиент-серверной архитектурой. Серверная часть (условно QaSystemCore) содержит в себе модули анализа текста

поступающих вопросов, классификатора и поиска ответов. Данные модули включают инструменты NLP и хранилище данных. В качестве клиентских приложений для разрабатываемой системы было решено использовать уже ранее разработанного чат-бота (поддерживающего соц. сети Вконтакте и Viber) и открытую систему обработки заявок OTRS.

Взаимосвязь между клиентскими компонентами и серверной частью основана на сервисно-ориентированной архитектуре (SOA).

QaSystemCore было решено реализовывать с помощью языка программирования Kotlin (версии 1.1) и фреймворка Spring (версии 4). Данная комбинация позволила относительно быстро разработать гибкое, надежное и защищенное программное обеспечение.

Из фреймворка были использованы следующие инструменты:

- Spring MVC - обеспечивает архитектуру паттерна Model - View - Controller (Модель - Отображение - Контроллер) при помощи слабо связанных готовых компонентов.
- Spring Data JPA – данная технология обеспечивает объектно-реляционное отображение простых JAVA объектов и предоставляет API для сохранения, получения и управления такими объектами.
- Spring Security – инструмент предоставляет механизмы построения систем аутентификации и авторизации, а также другие возможности обеспечения безопасности для корпоративных приложений.

В качестве хранилища данных была выбрана СУБД PostgreSql 10. Для корректировки компонентов БД использована библиотека Liquibase —независимая от базы данных библиотека с открытым исходным кодом для отслеживания, управления и применения изменений схемы базы данных.

В качестве инструмента для обработки естественного языка используется библиотека OpenNlp, содержащая в себе методы предобработки и классификации

русского текста.

Сервисно-ориентированная архитектура реализована с помощью веб-сервисов (Spring MVC включает в себя RESTful веб-сервисы), позволяющим передавать данные между компонентами в синхронном виде, и брокера сообщений RebbitMQ, предоставляющий асинхронный способ общения. Итоговая архитектура вопросно-ответной системы изображена на *рисунке 3*.

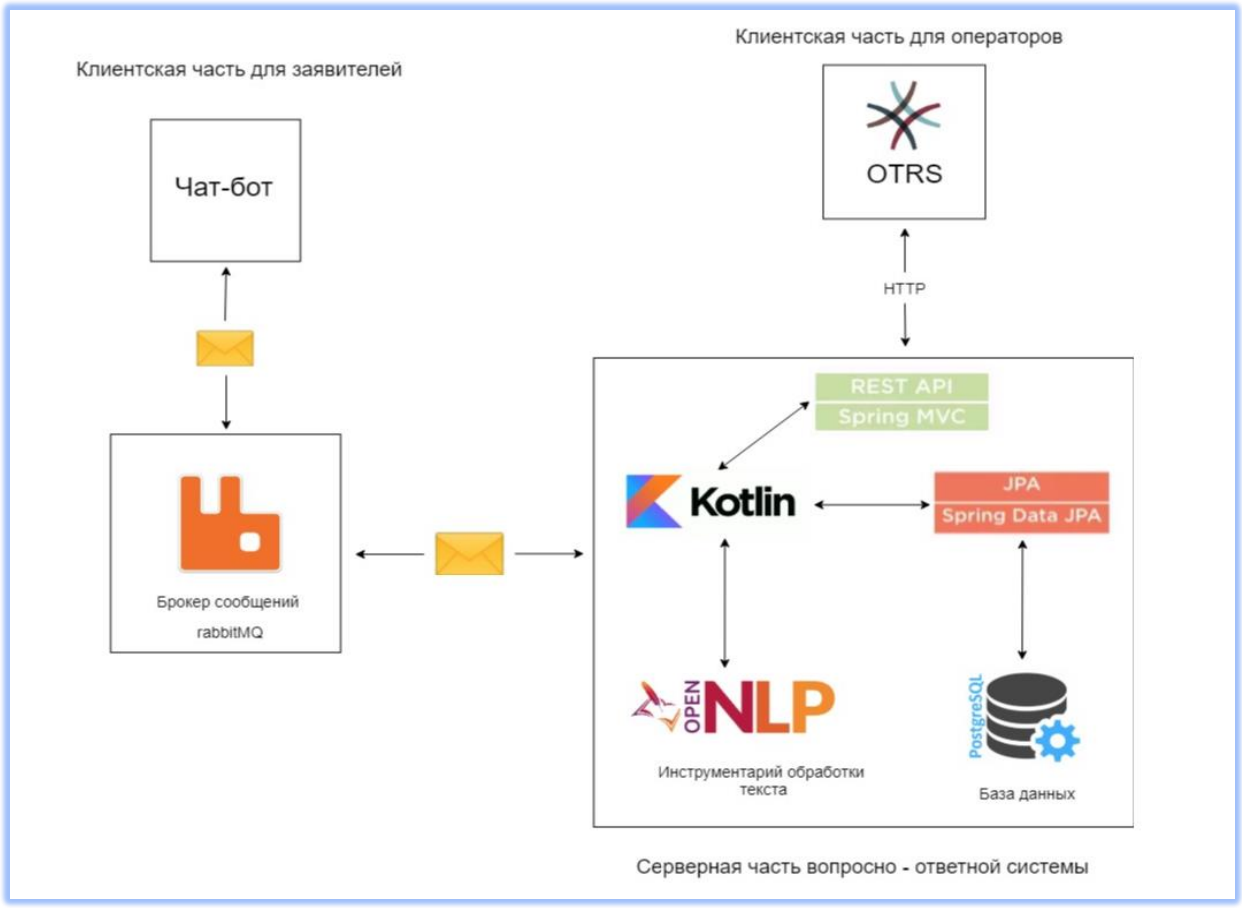

Рис. 3 – Архитектура вопросно-ответной системы

2.3 Этапы работы вопросно-ответной системы

Для того, чтобы провести консультацию пользователя, система выполняет следующие действия:

1) QaSystemCore получает вопрос от пользователя в текстовом виде от клиентской части (чат-бота) и анализирует его, а именно:

1. приводит символы текста в общий (нижний) регистр и удаляет знаки пунктуации;
2. над полученным текстом производит процессы лемматизации и токенизации;
3. передает полученный массив токенов классификатору, который определят наличие классов, проходящих условие заданного критерия поиска.

2) В зависимости от полученных результатов анализа текста, QaSystemCore выполняет следующие под-шаги:
   1. в случае, если модулю анализа текста удалость определить класс (или список классов) задаваемого вопроса, то система предлагает ответ (или список ответов) пользователю, который относится к данному классу (классам). После получения ответа пользователь может совершить следующие действия:
      I. если среди предложенных классов (вместе с предоставляемыми ответами) пользователь находит тот, который соответствует заданному вопросу, то он закрывает вопрос. Клиент передает пару вопрос-ответ QaSystemCore для того, чтобы система сохранила ее как верную, тем самым увеличив обучающую выборку для классификатора.
      II. если пользователь не нашел тематику предложенных ответов, которые соответствуют заданному вопросу, то вопрос перенаправляется оператору, который должен будет дать ответ в течение регламентированного срока.
   2. если анализ текста не выявил подходящие классы для задаваемого вопроса, то система перенаправляет вопрос оператору.
3) Ответ, данный оператором, перенаправляется пользователю. При этом система проверяет, если ли данный ответ в списке доверенных. После получения ответа, пользователю также предлагает выбор действий:
   1. Если пользователь удовлетворён ответом, то он закрывает вопрос. Также, если ответ относится списку доверенных, то система сохраняет пару вопрос-ответ как достоверную, и добавляет ее в обучающую выборку.

2. Если пользователь не удовлетворён ответом, или ему необходимо уточнить прочую информацию, касающуюся основного вопроса, то пользователь может задать новый, дополнительный вопрос, который будет перенаправлен оператору, миновав модуль анализа текста. После чего шаг 3 повторяет до того момента, пока пользователь не закроет вопрос (либо оператор не закроет его сама, посчитав его некорректным).

Основные этапы работы вопросно-ответной системы представлен на *рисунке 4.*

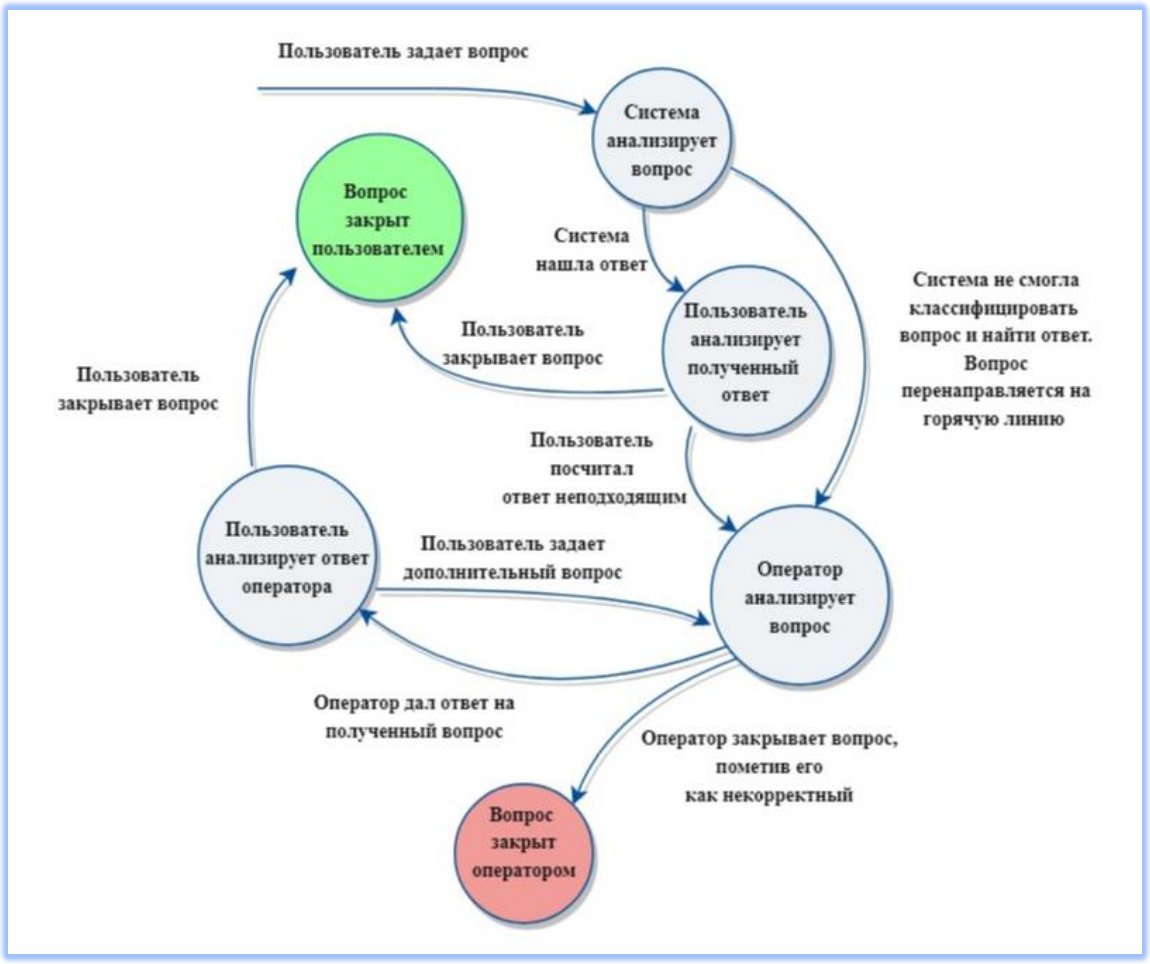

Рис. 4 – Основные этапы работы вопросно-ответной системы

2.4 Модуль обработки естественного языка

Предобработка текста происходит в несколько этапов:

1) с помощью регулярных выражений удаляются лишние пробелы, знаки препинания;
2) из текста удаляются стоп-слова;

3) текст приводится к единому, нижнему регистру;
4) с помощью библиотеки russianmorphology, представляющееся как открытое программное обеспечение [26], производится процесс лемматизации текса.
5) полученный результат обрабатывается инструментами для обработки естественного языка.

Проанализировав инструменты для обработки естественного языка, было решено использовать библиотеку OpenNLP.

Данная библиотека работает на платформе Java, что означает легкую интеграцию в уже готовые проекты, написанные под JVM. Также библиотека умеет работать с русским языком и имеет лицензию Apache 2.0.

Из библиотеки были задействованы инструменты токенизации и классификации текста. OpenNLP предоставляет следующие алгоритмы классификации:

- метод максимизации энтропии,
- наивная байесовская классификация,
- классификация на основе персептрона.

Чтобы определить оптимальный алгоритм был проведен эксперимент. Для этого были задействованы ранее собранные данные, содержащие в себе около 8000 вопросов и 160 ответов на них. Было определенно 160 классов, где каждый класс соответствовал определенному ответу («Необходимые документы для получения паспорта», «Действия, необходимые для получения СНИЛС» и т.п.). К каждом классу относятся от 20 до 150 вопросов. Над каждым вопросом была проведена процедура обработки текста. После чего, вопросы были разделены на две группы – обучающую (80% от всех вопросов) и проверочную (20% от всех вопросов).

Следующим шагом являлось построение моделей по обучающей выборке, используя разные параметры (разное количество итераций для алгоритмов максимизации энтропии и классификации на основе персептрона), и проведение проверки корректности их работы с использованием проверочной выборки.

Для более точной оценки эксперимент повторялся 10 раз для каждого случая, используя разные, случайно сформированные, обучающие и проверочные выборки. Средний процент точности работы моделей представлен в таблице 2.

Таблица 2

Средний процент точности работы моделей

Алгоритм / Количество итераций	Метод максимизации энтропии	Классификация на основе персептрона	Наивная байесовская классификация
10	67,47	50,32	73,65
100	78,96	52,82	
1000	81,15	51,59	
10000	79,81	51,43	

Среднее количество затраченного времени в секундах на обучение моделей представлено в таблице 3.

Таблица 3

Среднее количество затраченного времени на обучение моделей (в секундах)

Алгоритм / Количество итераций	Классификация на основе максимизации энтропии	Классификация на основе персептрона	Наивная байесовская классификация
10	1,2	0,7	17,2
100	9,8	6,1	
1000	98	61	
10000	979	608	

Опираясь на изученный материал и результаты эмпирического исследования, было решено использовать классификатор на основе метода максимальной энтропии.

2.5 Описание базы данных

В роли хранилища для тестовой выборки и другой необходимой информации для работы системы была выбрана реляционной базы данных PostgreSql, которая обладает следующими свойствами:

- управление структурированными и неструктурированными данным,
- возможность обрабатывать большие объемы данных,
- наличие многочисленных интерфейсов,
- механизмы, обеспечивающие отказоустойчивость,
- отрытый исходный код,
- кроссплатформенность.

Для хранения обучающей выборки используются две таблицы: train и category:

train – набор текстов, который используется при обучение модели классификации. В данной таблице хранятся вопросы, как в явном, так и в нормализованном, обработанном виде, вторичные ключи таблицы category и прочие данные, необходимый для обучения модели.

category – таблица, имеющая связь один ко многим к таблице train, хранит в себе достоверные, структурированные ответы, обновляющиеся при необходимости, а также их краткое описание, что, само по себе, является классом вопросов.

Для того, чтобы реализовать диалог между системой и пользователем, были созданы 2 таблицы – ticket и question, которые хранят в себе перенаправленные вопросы, предоставленные ответы, а также идентификаторы записей в клиентских системах.

ticket – после того, как пользователь задает вопрос, в данной таблице создается запись, хранящая в себе время создания вопроса, время закрытия, id записи в клиентской части для пользователей(чат-бота) и прочие данные, характеризующие статус диалога между пользователем и системой.

consultation – в данной таблице хранятся вопросы, время их создания, ответы, время предоставления ответа, идентификатор записи (articleOtrsId) в клиентской части операторов (OTRS), после того, как вопрос переправлен оператору, идентификатор отвечавшего оператора, статус вопроса и прочие данные.

Связь между таблицами ticket и consultation определенна как один ко многим – в рамках одного диалога пользователь может задать несколько вопросов. Данные таблицы, в отличие от таблиц train и category, не используются в модуле обработки естественного языка.

Доступ к таблицам осуществляется с помощью инструментов Spring Data JPA. Для этого в приложение были созданы интерфейсы, наследующиеся от интерфейса JpaRepository, предоставляющего базовый набор методов работы с БД. Названия интерфейсов соответствуют названиям таблиц: TicketDao, ConsultationDao, TrainDao, CategoryDao. Каждый из интерфейсов представляет собой репозиторий, расширяющий список методов интерфейса JpaRepository. Также были созданы сущности (entity), реализующие отображение (maps) с реляционными таблицами БД: Train, Category, Ticket, Consultation.

Интерфейс TicketDao позволяет работать с таблицей ticket, реализуя следующие методы:

- findById – находит и возвращает сущность Ticket по ее идентификатору, позволяя проводить работу с связанной с ней записью в таблице ticket, получать или изменять данные. В качестве параметра метод принимает первичный ключ (id) в виде uuid;
- findByUserId – метод, аналогично методу findByTicketId, возвращает список сущностей Ticket. Поиск осуществляется по id клиента, который создал диалог;

- findByStatus (userId: UUID): возвращает список сущностей Ticket. Поиск осуществляется по элементам перечисления, соответствующим статусу диалога (OPEN, CLOSED, PROCESSING);
- updTicketStatusByTicketId – изменяет статус определенной записи таблицы ticket по ее id.

Интерфейс ConsultationDao позволят работать с вопросами пользователей из таблицы consultation. Так как таблицы ticket и consultation имеют связь один ко многим, доступ к записям таблицы consultation можно получить с помощью интерфейса TicketDao, найдя определенную сущность Ticket по ее идентификатору или реализовав дополнительные методы поиска. Но создание отдельного интерфейса ConsultationDao для таблицы consultation способствовало читаемости кода и формированию менее ресурсоемких запросов. Интерфейс содержит в себе следующие методы:

- findById – находит и возвращает сущность Consultation по ее идентификатору;
- findByArticleOtrsId – возвращает сущность Consultation по идентификатору записи, полученному со стороны клиентской части операторов;
- findByStatus – возвращает список сущностей Consultation. Поиск осуществляется по элементам перечислений, соответствующим статусу вопроса (OPEN, CLOSED, PROCESSING);
- findByQuestionDate – возвращает список сущностей Consultation. Поиск осуществляется по дате создания записи в таблице consultation;
- updConsultationStatusById – изменяет статус определенной записи таблицы consultation по ее идентификатору;
- updConsultationStatusByArticleOtrsId изменяет статус определенной записи таблицы consultation по идентификатору записи, полученному со стороны клиентской части операторов.

Методы интерфейса TrainDao:

- findById – находит и возвращает сущность Train по ее идентификатору;
- findByOriginalPhrase – возвращает сущность Train. Поиск осуществляется по тексту вопроса;
- findByNormalizedPhrase – возвращает сущность Train. Поиск осуществляется по обработанному тексту вопроса;

Методы интерфейса category:

- findById – находит и возвращает сущность Category по ее идентификатору;
- findByCategory – находит и возвращает сущность Category по ее названию;
- findByAnswer – находит и возвращает сущность Category. Поиск осуществляется по тексту ответа, хранящемуся в данной категории;
- updActiveById – изменяет статус активности категории по ее идентификатору.

Добавления новых записей и изменения старых происходит с помощью стандартных методов интерфейса JpaRepository.

2.6 Взаимодействие клиентских приложений с сервером

Так как решено было реализовывать систему на основе сервисно-ориентированной архитектуре, необходимо было определить способ передачи данных между клиентами и разрабатываемой системой. Помимо использования RESTful веб-сервисов было решено использовать брокер сообщений RabbitMQ. Наличие брокера сообщений гарантирует системе независимость между ее компонентами друг от друга, экономию ресурсов, отказоустойчивость и гарантию последовательной обработки. Также, это позволяет реализовать асинхронное взаимодействие между участниками системы, необходимое в случае, когда система не смогла автоматически определить подходящий ответ, и вопрос был перенаправлен на рассмотрение в горячую линию операторам.

Для клиентов (чат-бот и OTRS) с помощью RESTful были реализованы веб-сервисы, основными которые являются:

- **askQuestion –** принимает все необходимые для анализа данные (вопрос, id клиента, тип вопроса (основной или повторяющийся) и т.п.). Синхронно возвращает ответ (список ответов) в случае, если удалось классифицировать вопрос и подобрать ответ. В обратном случае, возвращает идентификаторы записей из таблиц ticket и consultation**.** Предназначен для клиента, инициализирующий диалог – чат-бота.
- **ackAnswer** – метод, потверждающий получение ответа. Передает данные, включая тег действия (перенаправить вопрос в горячую линию, закрыть вопрос без действия или закрыть вопрос с сохранением пары вопрос-ответ в обучающей выборки или) – также предназначен для клиента, инициализирующий диалог.
- **passAnswer** – метод, с помощью которого QaSystemCore получает ответ от оператора. Предназначен для обмена данными с OTRS.

В свою же очередь на стороне OTRS также пришлось реализовывать REST веб-сервисы, такие как CreateTicket и UpdateTicket, а также функцию обратного вызова к ackAnswer, которые позволили обмениваться данными с QaSystemCore [27].

Для асинхронного обмена данными между чат-ботом и QaSystemCore в брокере сообщений были созданы следующие каналы обмена (exchanges)

- **ask_question_events** – аналог синхронного askQuestion. После обработки вопросов ответы от QaSystemCore будут переданы в канал send_answer_events. Содержание ответов также зависит от работы модуля классификации.
- **send_answer_events** – в данном канале хранятся ответы от QaSystemCore.
- **ack_request_events** – аналог веб-сервиса ackAnswer.
- **ack_response_events** – в данном канале хранятся подтверждения на запросы ack_request_events.

2.7 Пример работы вопросно-ответной системы

На *рисунках 5* и *6* представлен сценарий работы, в котором вопросно-ответная система автоматически предоставляет ответ пользователю на заданный вопрос.

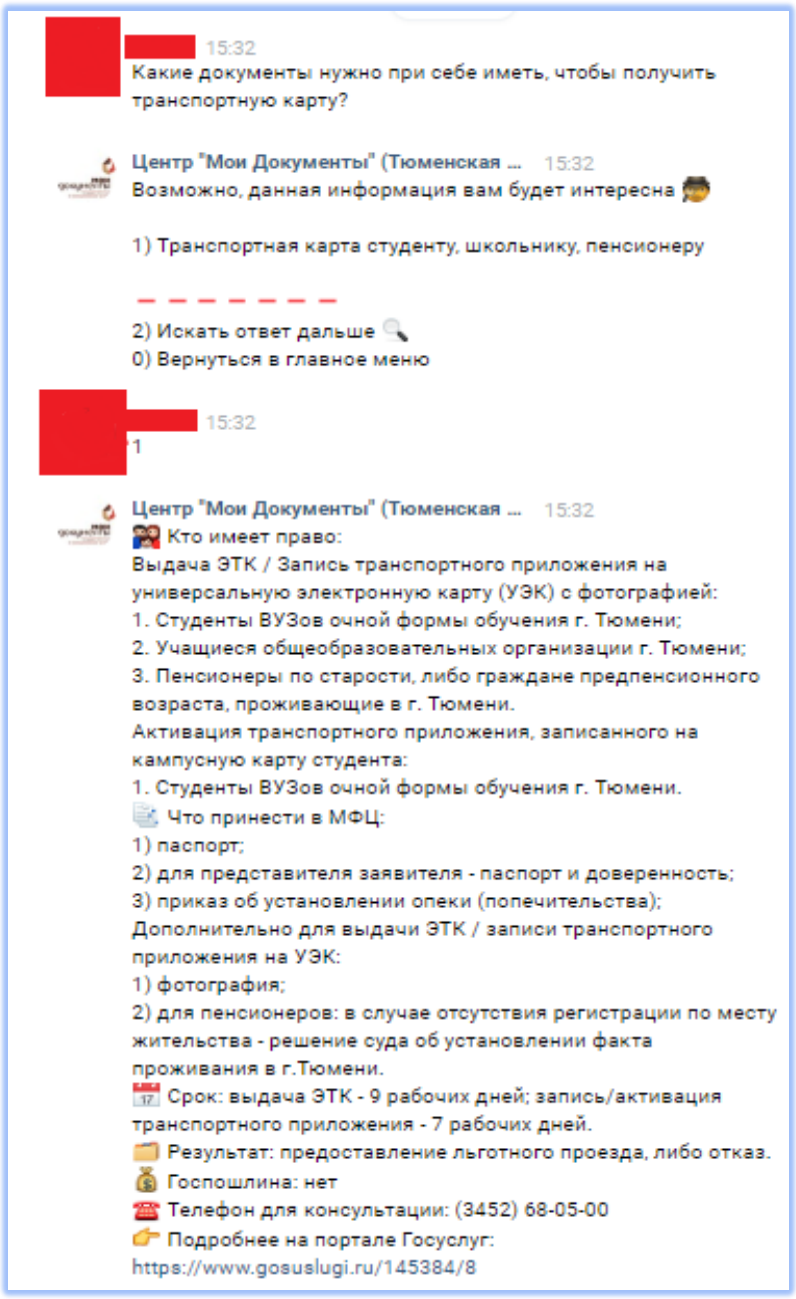

Рис. 5 – Интерфейс чат-бота. Предоставление ответа пользователю

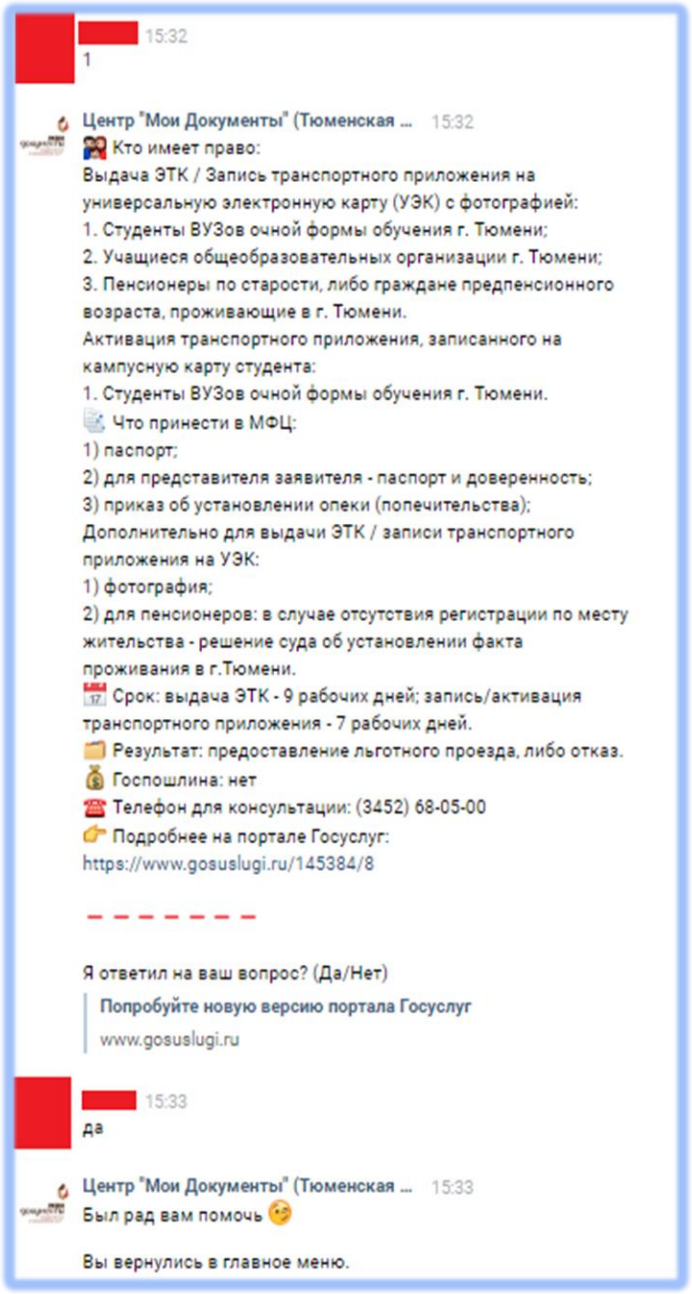

Рис. 6 – Интерфейс чат-бота. Пользователь закрывает вопрос

После завершения диалога система сохранила пару, состоящую из заданного вопроса и выбранного ответа, как достоверную. Данная информация будет учтена во время следующего этапа переобучения модели классификации.

В случае, если пользователь не удовлетворён предоставленным ответом, то он может перенаправить его оператору, который предоставит ответ пользователю в течение регламентированного срока (*рисунок 7)*.

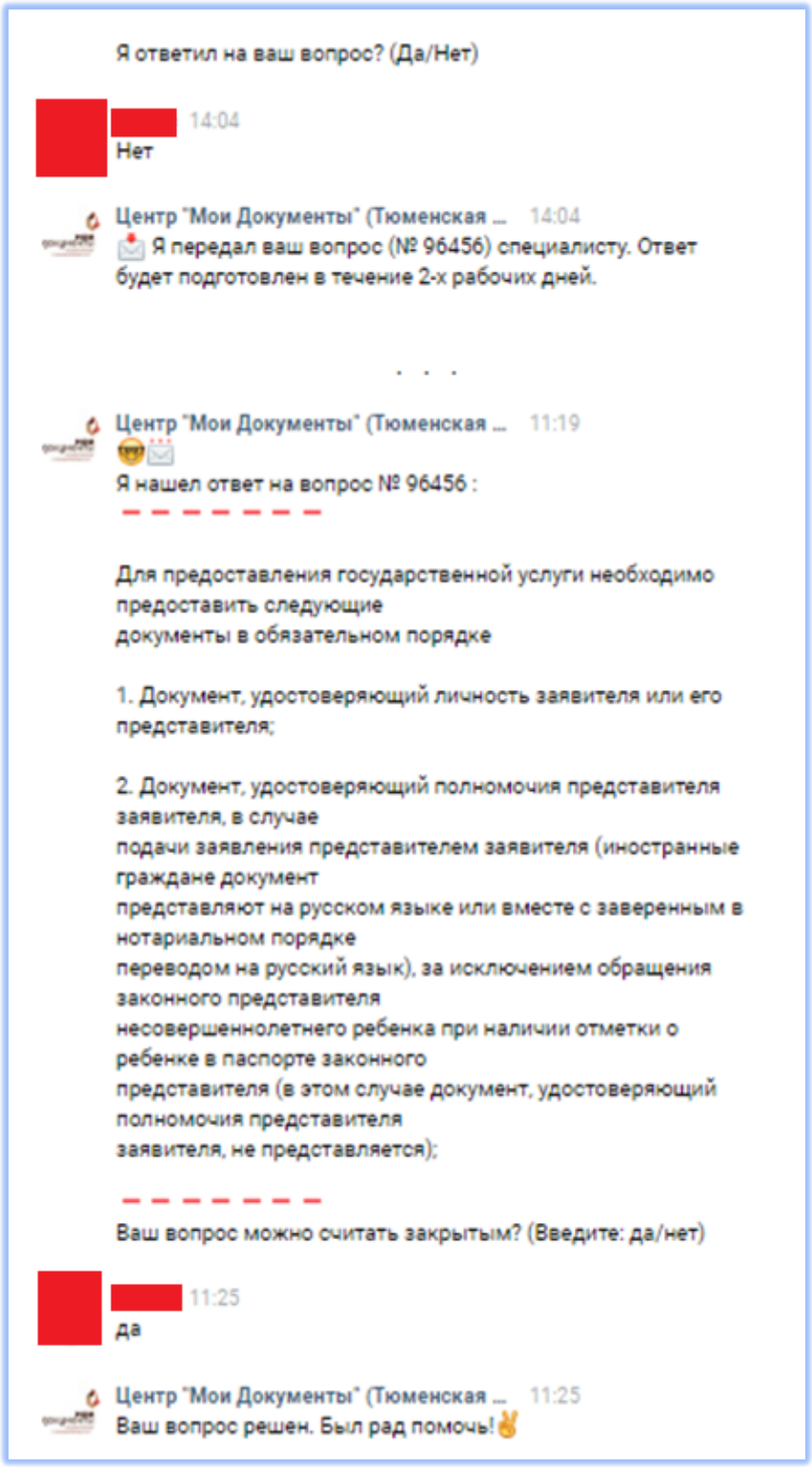

Рис. 7 – Интерфейс чат-бота. Пользователь закрывает вопрос, который ранее был перенаправлен на горячую линию

Заключение

В результате проделанной работы выполнен анализ подходов к решению задачи автоматического поиска ответа на вопрос, сформулированного пользователем на естественном языке. Были рассмотрены работы отечественных специалистов и исследователей в области разработки вопросно-ответных систем. Были изучены типы вопросно-ответных систем, их архитектуры, основные модули, а также рассмотрены основные проблемы, связанные с реализацией таких систем.

Рассмотрены наиболее популярные алгоритмы классификации, их достоинства и недостатки. Также удалось изучить инструменты анализа естественного языка, такие как Stanford CoreNLP, OpenNLP, NLTK, LingPipe и GATE.

На этапе разработки была спроектирована и реализована вопросно-ответная система. Во время реализации системы были задействованы следующие технологии: Kotlin, Spring, PostgresSql, RabbitMQ. С помощью сервисно-ориентированного подхода к системе удалось интегрировать в роли клиентских приложений систему заявок OTRS и чат - бота. Удалось реализовать модуль анализа вопроса с помощью библиотеки OpenNLP. Также была собрана выборка для обучения и тестирования классификатора, основанного на алгоритме максимизации энтропии.

Список литературы

1) Green Jr., Bert F; et al. "Baseball: an automatic question-answerer". Western joint IRE-AIEE-ACM computer conference — 1960 — [Электронный ресурс] URL: https://web.stanford.edu/class/linguist289/p219-green.pdf (дата обращения: 15.06.2019)
2) Woods, William A; Kaplan, R. "Lunar rocks in natural English: Explorations in natural language question answering". Linguistic structures processing 5.5 — 1977.
3) Apple - Siri - Frequently Asked Questions., Apple Inc — [Электронный ресурс] URL: https://www.apple.com/siri/ (дата обращения: 15.06.2019).
4) WloframAlpha, Wolfram Research, — [Электронный ресурс] URL: https://www.wolframalpha.com/ (дата обращения: 15.06.2019).
5) Алиса. Голосовой помощник от компании Яндекс, Яндекс — [Электронный ресурс] URL: https://alice.yandex.ru (дата обращения: 15.06.2019).
6) Науменко А. М., Шелудько С. Д., Юлдашев Р. Ю., Хлебников Н. О. Разработка вопросно-ответной системы с использованием машинного обучения // Молодой ученый. — 2017. — №8. — С. 36-40. — [Электронный ресурс] URL: https://moluch.ru/archive/142/40056/ (дата обращения: 15.06.2019)
7) Лапшин В. А. Вопросно-ответные системы: развитие и перспективы //Научно-техническая информация. Серия 2: Информационные процессы и системы. – 2012. – №. 6.
8) Мочалова А. В., Мочалов В. А. Интеллектуальная вопросно-ответная система //Информационные технологии. – 2011. – №. 5.
9) Медовый В. С. Русскоязычная вопросно-ответная система как система формального вывода// Медицинские компьютерные системы (МЕКОС), Москва – 2017 — [Электронный ресурс] URL: http://www.dialog-21.ru/media/3975/medovyyvs.pdf (дата обращения 15.06.2019)
10) Бухаров М. Н. Автоматизированное консультирование на основе гибридного интеллекта //Сборник научных трудов. – 2012.

11) Кулешов А. С., Беляев С. А. Формирование вопросно-ответной системы в условиях ограниченного объема семантически размеченного корпуса //Программные продукты, системы и алгоритмы. – 2016. – №. 4.
12) Ненаусников К. В., Кулешов С. В., Зайцева А. А. Анализ подходов к созданию базы данных вопросно-ответных систем на основе автоматической обработки естественно языковых текстов // Информационные технологии и телекоммуникации. 2018. Том 6. № 1. С. 92–100. — [Электронный ресурс] URL: http://www.sut.ru/doci/nauka/review/20185/92-100.pdf (дата обращения: 15.06.2019)
13) Соловьёв А. А., Пескова О. В. Об архитектурах программных систем вопросно-ответного поиска //Научно-техническая информация. Серия 2: Информационные процессы и системы. – 2013. – №. 10.
14) Burger, J., Cardie, C., Chaudhri, V., Gaizauskas, R., Harabagiu, S., Israel, D., Jacquemin, C., Lin, C-Y., Maiorano, S., Miller, G., Moldovan, D., Ogden, B., Prager, J., Riloff, E., Singhal, A., Shrihari, R., Strzalkowski, T., Voorhees, E., Weishedel, R. Issues, Tasks and Program Structures to Roadmap Research in Question Answering (QA) — 2001. — [Электронный ресурс] URL: http://www-nlpir.nist.gov/projects/duc/papers/qa (дата обращения: 15.06.2019)
15) Мочалова А.В. Семантический анализатор русскоязычного текста для вопросно-ответной системы: диссертация кандидата Технических наук: 05.13.18 / Мочалова Анастасия Викторовна; 2017 - 128 с
16) .Lee Y.-S., Santorini B. Towards resolving Webelhuth’s paradox: evidence from German and Korean / eds. Koster J., van Reimsdijk H. – 1994. – P. 257 – 300.
17) Nedjalkov I. Evenki. – London ; New-York : Routledge, 1997.
18) Казакевич О. А., Митрофанова Н. К., Рудницкая Е. Л. Истории жизни автохтонного населения Сибири: публикации глоссированных текстов (публикация 2-я) // Вестник РГГУ. Сер. «Языкознание». – 2009. – Т. 11, № 6
19) Chang S.-J. Korean. – Amsterdam ; Philadelphia : John Benjamins – 1996

20) Hornik K. openNLP: Apache OpenNLP tools interface //R package version 0.2-5. – 2015. — [Электронный ресурс] URL: https://opennlp.apache.org/ (дата обращения: 15.06.2019)

21) Cunningham H., Maynard D., Bontcheva K. and Tablan V. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications (In proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics, 2002) (англ.) // University of Sheffield. — 2009. — [Электронный ресурс] URL: https://gate.ac.uk/sale/acl02/acl-main.pdf (дата обращения: 15.06.2019)

22) Риз Р. Обработка естественного языка на Java //ДМК-Пресс.– 2016.

23) Кафтанников И. Л., Парасич А. В. Особенности применения деревьев решений в задачах классификации //Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. – 2015. – Т. 15. – №. 3.

24) Баев И. О. Использование метода опорных векторов в задачах классификации //Международный журнал информационных технологий и энергоэффективности. – 2017. – Т. 2. – №. 2.

25) Соловьёв А. А., Пескова О. В. Построение вопросно-ответной системы для русского языка: модуль анализа вопросов //Новые информационные технологии в автоматизированных системах. – 2010. – №. 13.

26) Kuznetsov A. // Russian Morphology for Apache Lucene — [Электронный ресурс] URL: https://github.com/AKuznetsov/russianmorphology (дата обращения: 15.06.2019)

27) OTRS AG //OTRS Documentation — [Электронный ресурс] URL: https://doc.otrs.com/doc/manual/admin/stable/en/content/processes-automation/web-services.html (дата обращения: 15.06.2019)

Приложение А

Инструменты обработки и анализа текста

Названия / Характеристик	**CoreNLP**	**LingPipe**	**GATE**	**OpenNLP**	**NLTK**
Платформа	Java	Java	Java	Java	Python
Лицензия	GPL	Коммерческая и некоммерческая	LGPL	Apache License	Apache License
Языки	английский, китайский	английский	английский испанский, китайский, арабский, болгарский, французский, немецкий, хинди, итальянский, румынский, русский	английский, русский	английский, русский

Возможности	Графематический анализ (эвристика)	Извлечение именованных сущностей (машинное обучение), средства для многопоточной работы, тесты	Графематический анализ (регулярные выражения, машинное обучение)	Графематический анализ (машинное обучение)	Синтаксический анализ, Морфологический анализ, Графематический анализ (регулярные выражения, машинное обучение)