

Московский государственный университет имени М. В. Ломоносова Факультет вычислительной математики и кибернетики

Кафедра алгоритмических языков

Гончаренко Евгений Евгеньевич

Вопросно-ответная система по языку Питон

выпускная квалификационная работа

Научный руководитель:

к.ф.-м.н., доцент Головин Игорь Геннадьевич

Москва, 2023

Аннотация

В последнее время все больше крупных компаний выделяют свои ресурсы на создание искусственных диалоговых помощников (Алиса от Яндекса, Ассистенты Салют от Сбер и др). С такими системами можно поддерживать достаточно информативный диалог. Ассистенты умеют выполнять простые команды: ставить таймер или будильник, вызывать такси, управлять умным домом. Но в то же время разработка таких систем достаточно затратный процесс и требует ресурсы на его поддержку. В большинстве случаев не требуется, чтобы система умела поддерживать диалог, а требуется ответ на конкретный вопрос. Аналог современных вопросно-ответных систем называются экспертными системам. Такая система включает в себя оболочку на естественном языке и позволяет задавать вопросы на узкую тематику. При этом возможно не акцентировать такой диалог под решение специализированной задачи.

В данной работе был предложен процесс создания вопросно-ответной системы по языку Питон, позволяющий построить подобную систему с возможностью получения кратких ответов на естественном языке. Подход также был реализован в виде программного средства с использованием современных моделей машинного обучения, таких как BERT и приведен реальный пример практического использования.

Содержание

1. [Введение 4](#_TOC_250023)
2. [Постановка задачи 5](#_TOC_250022)
3. Обзор архитектур существующих решений
   1. [Формирование ответа на вопрос на естественном языке 12](#_TOC_250011)
4. [Построение решения задачи 14](#_TOC_250010)
   1. [Структура информации в StackExchange 14](#_TOC_250009)
   2. [Извлечение информации из базы данных StackExchange 15](#_TOC_250008)
   3. Формирование ответа на заданный вопрос на естественном языке — BERT . 17
   4. [Извлечение релевантного документа 20](#_TOC_250007)
   5. [Объединение этапов 21](#_TOC_250006)
5. [Заключение 30](#_TOC_250000)
6. **Список литературы 35**
7. Введение

Современный мир невозможно представить без информационных технологий. Общение в социальных сетях, поиск информации в интернете. Все направления активно развиваются, появляются новые задачи, а объем информации только увеличивается во всемирной паутине.

В связи с этим появляется такой важный критерий при решении той или иной информации – скорость нахождения нужной информации. А что такое нужная информация? Предположим, что мы делаем той или иной запрос в интернете. И нам важно найти наиболее релевантный ответ, смысловая нагрузка которого нам будет подходить больше всего.

Поиск требуемой информации в поисковой системе не всегда наиболее удобный, если мы рассматриваем более узконаправленную нишу, нам не удобно переходить по десяткам гиперссылок на веб-сайты, ища необходимое по ключевым словам. Более того, далеко не всегда ключевые слова несут истинную смысловую нагрузку запроса.

За последнее десятилетие можно заметить тенденцию в интеграциях диалоговых систем с чат-ботами, которые могут помочь ответить на интересующий вопрос в узкой нише: сервис такси, мобильное приложения банка и т.п. В таком случае как раз качественно разработанные QA-системы помогут сэкономить время на поиск информации.

Но всегда ли у нас возникают подобные трудности только при пользовании того или иного продукта? Примером запроса может являться вопрос по устройству того или иного языка программирования, понимания алгоритмов или навыка использования специфичной библиотеки, используемой при разработке. В таком случае первое, что приходит в голову – это использование таких ресурсов как StackExchange. Но всегда ли нам удобно его использовать? Иногда для поиска необходимого ответа может потребоваться посещение десяток страниц, просмотр сотен ответов на задаваемый вопрос. Безусловно помогает сортировка по полезности таких ответов, но тем не менее зачастую на это уходит много времени. А что если, мы мысленно представим у себя в голове, что рядом с нам есть наставник-консультант? Человек, который может незамедлительно ответить на интересующий для нас вопрос и предложить надежное решение? Именно в роли такого помощника и выступает QA-система.

В данной работе предложен подход, который позволяет в формате диалоговой системы распознавать запрос на естественном языке и в качестве результата выдавать наиболее релевантные ответы, построенные на организованном поиске по базе данных StackExchange, по языку Питон. Таким образом, конечный пользователь может задавать любой вопрос в формате диалога по интересующей его тематике и получать лучшие ответы, собранные из наиболее полезных советов, комментариев и фрагментов программного кода, что может помочь существенно сэкономить время на поиск той или иной информации.

1. Постановка задачи

Целью работы является разработка системы поиска информации по базе данных StackExchange, позволяющей задавать вопросы на естественном языке и использовать информацию для получения наиболее релевантных ответов. При формировании ответа, система должна уметь отвечать на естественном языке, а также при необходимости дополнять свой ответ релевантными фрагментами кода с соответствующей информацией.

1. Обзор архитектур существующих решений

Для определения архитектуры решения необходимо понять, а как система будет использоваться. Локально на стороне клиента или облачно, в формате отправки запросов на сервер. От этого зависит, можно ли делать монолитную архитектуру с локальным хранением и обновлением базы данных, или же основанной на микросервисной архитектуре с хранением баз данных на удаленном сервере. В нашем случае мы разрабатываем вопросно-ответную систему. RPS – количество запросов в виде пользователей большое, приложение должно занимать разумно малый размер по памяти и система должна быть отказоустойчивая. База данных имеет большой размер, а ее содержание постоянно обновляется и актуализируется. По этой причине хранить ее целесообразно в одно месте на удаленном сервере, а монолитная архитектура отпадает, так как намного проще использовать уже готовые сервисы по контейнеризации системы и ее масштабированию.

Вопросно-ответная система должна:

1. Принимать заявки из заданных каналов связи (чат на веб-сайте, мобильное приложение с возможностью отправки сообщений на сервер).
2. Производить классификацию запроса на заданные тематики, соблюдая SLA < 2 секунды на ответ.
3. Иметь интуитивно понятный интерфейс.
4. Записывать логи своей работы для системы мониторинга и дебаггинга.

Разберем подробнее каждый из пунктов.

#### Вопросно-ответная система должна уметь принимать и обрабатывать заявки из заданных каналов связи пользователей.

Первое. Для упрощения архитектуры и поиска решения пользователи задают свои вопросы в текстовом формате, в произвольной форме. При том условии, что конечный запрос пользователя является некоторой формулировкой того или иного задаваемого вопроса. А вопросно-ответная система в свою очередь должна уметь принимать и обрабатывать такие заявки – понимать смысл задаваемого вопроса.

Второе. Классификатор является важным компонентом вопросно-ответной системы. Алгоритм классификации ресурсозатратный и требует значительного потребления GPU, что стоит учитывать при проектирования веб-сервера.

Третье. Упрощенная система логирования строится благодаря функционалу внутренних возможностей PostgreSQL. Также для автоматизации процессов логирования есть специальные фреймворки, такие как: log4j, log4net, Retrace, Logback, Logstash и другие, что весьма упрощает решение этой задачи.

Решаемую задачу можно разделить на четыре разных составляющих:

* 1. Распознавание сформулированного вопроса на извлечение его смысловой нагрузки.
  2. Поиск подходящих записей в базе данных, отсортированных по релевантности, с учетом их связей с дополнительной информацией.
  3. Объединение и обработка найденной информации;
  4. Формирование и отправка ответа на заданный вопрос на естественном языке с возможной дополнительной информацией.

Система в общем виде состоит из 4 компонентов, а именно:

1. Server – микросервис для обработки текстовых сообщений и их классификации.
2. ApiGateway – микросервис для маршрутизации всех запросов в системе между микросервисами.
3. База данных PostgreSQL, которая содержит всю необходимую информацию для работы системы, а также логи ее работы.
4. Сайт для возможности удобного использования системы.
   1. Формирование ответа на вопрос на естественном языке

Обработка естественного языка (Natural Language Processing, NLP) – это область вычислительной лингвистики, ориентированная на разработку машин, способных понимать человеческие языки. В современной компьютерной лингвистике понимание смысла написанного или сказанного достигается с помощью векторных моделей естественного языка.

В результате анализа за последние годы со значительным отрывом от своих предшественников в задачах распознавания естественного языка побеждает векторизатор предложений [NLU] на базе архитектуры BERT от компании Google. Именно он в данный момент применяется в соответствующей поисковой системе для улучшения понимания 10% запросов на английском языке. Преимуществом этой нейронной сети перед остальными является то, что она использует ненаправленное обучение для языковых моделей, т.е. позволяет понимать контекст слова на основе окружающих его слов, а не только на основе слов с одной стороны предложения.

Так как это решение является наиболее эффективным для получения ответа на вопросы на естественном языке, имеет смысл попробовать его применить и к решаемой задаче на одном из этапов. Это, в свою очередь, приводит к проблеме преобразования программного кода и артефактов процесса его разработки в некоторое подобие текста на естественном языке.

1. Построение решения задачи

Решаемую задачу можно разделить на несколько стадий, для каждой из которых необходимо построить решение. Следует также помнить, что артефакты разработки программного обеспечения могут поступать из множества источников в различных формах, поэтому решение должно быть легко расширяемым.

* 1. Структура информации в StackExchange

Из чего строится основная информация на данном ресурсе? В основе сайта лежат категории задаваемых вопросов пользователями: языки программирования, встроенные библиотеки, решения типовых практических задач и т.д. Далее для каждой категории идут списки вопросов. Каждый вопрос – отдельная страница, на которой идут обсуждения пользователей (их ответы) с возможными описаниями и вставками кода. Вся эта информация хранится в формате xml блоков, что позволяет вполне удобно распознавать и получать необходимую информацию при парсинге.

**<row** Id**=**"413146"PostTypeId**=**"1"AcceptedAnswerId**=**"413162"CreationDate**=**"2015-04-01T15:44:14.107"Score**=**"0"ViewCount**=**"333"Body**=**"&lt;p&gt;У меня стоит задача зашифровать текст. Я считываю ..."OwnerUserId**=**"176241"LastEditorUserId**=**"15146"LastEditDate**=**"2015-05-14T10:11:08.220"LastActivityDate**=**"2015-05-14T10:11:08.220"Title**=**"Использование строки типа String^"Tags**=**"&lt;c++-cli&gt;"AnswerCount**=**"2"**>**

Рис. 1: Формат исходной информации

Из ключевых тегов можно подчеркнуть:

* 1. PostTypeId – необходим для связи задаваемого вопроса и соответствующих ответов
  2. Tags – для понимая к какой категории относится данное обсуждение
  3. Score – оценка того, на сколько тот или иной ответ полезен (необходим для сортировки по релевантности)
  4. Titile – соответствующие заголовки вопросов или ответов
  5. Text – более развернутые текстовые описания
  6. AnswerCount – количество ответов на задаваемый вопрос (чем больше было ответов и обсуждений, тем можно сделать вывод о степени актуальности вопроса)

Файлы с ключевой информацией:

1. Posts.xml – информация с всеми задаваемыми вопросами
2. Comments.xml – ответы для каждого задаваемого вопроса
3. PostLinks.xml – связи между вопросами по однотипным проблемам

Опираясь на эту структурированную информацию, мы можем уже формировать нашу будущую базу данных, в которой ключами будут служить сами вопросы, а телом – соответствующие наиболее релевантные ответы с дополнительной информацией с учетом всех нюансов и связей с аналогичными вопросами.

* 1. Извлечение информации из базы данных StackExchange

В качестве целевого языка для исследования поставленной задачи был выбран язык Питон. Это обосновано тем, что Питон является одним из самых простых для разработки, широко функциональным для больших проектов, в том числе для серверной разработки и парсинга данных. Также язык Питон является объектно-ориентированным языком программирования, т.е. в его основе лежит понятие объекта — сущности, объединяющей в себе данные и методы. Это ведёт к понятной и предопределённой структуре исходного кода, а следовательно, к большим возможностям для разработки и его чтения, что тоже имеет важную роль, ведь недостаточно единоразово написать текст программы, важно иметь возможность быстрой поддержки и модернизации.

Таким образом, весь исходный код можно представить как совокупность методов, имеющих реализацию, и уже на их основе строить парсинговую систему.

Далее рассмотрим более подробно структуру нашего парсера. Цель: извлечение необходимой информации из xml файлов и порождение SQL запросов для записи собранной информации в нашу PostgreSQL базу данных.

Основные классы:

1. Users – информация о пользователях
2. Posts – информация со всеми задаваемыми вопросами
3. PostLinks – связи между задаваемыми вопросами
4. PostHistory – история обновлений и дополнений конкретных обсуждений
5. Comments – непосредственно ответы на вопросы
6. Tags – информация о том какой вопрос к какой теме относится

Основные методы – приведения различных типов данных к валидному формату для базы данных.

В главной функции main() через средства асинхронной работы мы создаем пулл будущих запросов в базу данных, открываем соединение с нашим сервером, создаем чанки и совершаем загрузку xml файлов.

class ClassName:

QUERY = """INSERT INTO users VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, $11, TO\_TIMESTAMP($12, 'YYYY-MM-DDTHH24:MI:ss'), TO\_TIMESTAMP($13, 'YYYY-MM-DDTHH24:MI:ss'))"""

FILE\_NAME = "json/Users.json"

async def make\_request(db\_pool, line):

await db\_pool.fetch(

Users.QUERY,

try\_to\_get\_int\_value(line, 'Id'),

try\_to\_get\_int\_value(line, 'AccountId'),

…

)

await sleep(.1)

Рис. 2: Структура класса

Таким образом мы получили структурированную и связанную базу данных для дальнейшей работы нашей вопросно-ответной системы.

* 1. Формирование ответа на заданный вопрос на естественном язы ке — BERT

Как уже было замечено в обзоре существующих решений, нейронная сеть BERT от компании Google является одним из лучших решений на данный момент в задачах распознавания естественного языка. Так как стоит задача извлечения краткого ответа при возможности, целесообразно попробовать применить BERT при решении данной проблемы.

BERT основана на модели трансформер и использует двунаправленное обучение (или, как замечают сами разработчики, ненаправленное обучение), позволяя понимать контекст слова на основе соседних слов с обоих сторон — существовавшие до этого модели анализировали последовательность лишь слева направо или справа налево. Достигается это за счёт того, что модель обучают на основе предложений с замаскированными словами. BERT также способна определять, взаимосвязаны ли два предложения между собой.

Единственным препятствием для применения BERT в решаемой задаче является то, что она рассчитана на работу со связным текстом на естественном языке. Из этого вытекает то, что на основе фрагмента кода необходимо сгенерировать некоторый текст.

* 1. Извлечение релевантного документа

Если применять модель BERT к каждому документу, то возникает две существенные проблемы. Во-первых, BERT вернёт ответ для каждого с той или иной вероятностью, но т.к. документы строятся по одному шаблону, эта вероятность будет мало говорить о результате. Например, на вопрос категории BERT с высокой долей вероятности вернёт для каждого документа содержащуюся в нём дату, но это не будет значить, что документ соответствует запросу. Во-вторых, применение модели BERT к каждому документу, которых может быть значительное количество, было бы довольно ресурсозатратно и долго. Таким образом, перед извлечением ответа на поставленный вопрос необходимо сузить диапазон поиска, т.е. найти некоторое количество релевантных документов.

При нормализации токенов запроса и документа, чтобы нивелировать точное совпадение целесообразнее использовать FastText — библиотеку от компании Facebook, позволяющую с помощью обучения моделей нейронных сетей получать векторные представления слов таким образом, что близкие по смыслу слова находятся близко и в векторном пространстве. В общем случае такие алгоритмы называются Word2vec.

Основной принцип строится на том, что получив векторные представления запроса и каждого сгенерированного документа, можно оценить их близость в векторном пространстве с помощью косинусного расстояния:

cos *θ* =

*AB*

*ǁAǁ*2*ǁBǁ*2

Для представления документа в виде вектора используется нормализованная сумма векторов его слов.

Перед применением векторизации из запросов и документов удаляются знаки препинания и стоп-слова, вносящие шум.

* 1. Объединение этапов

Объединяя описанные выше этапы, итоговое решение можно описать следующим алгоритмом:

1. Загружается база данных;
2. Анализируется и собирается информация по каждому вопросу
3. Для каждого вопроса строится документ, представляющий собой текст на естественном языке, объединяющий информацию ответов и программного кода;
4. Для каждого документа удаляется пунктуация и стоп-слова, затем строится векторное представление;
5. Для поступающих запросов строится векторное представление;
6. Для векторных представлений запроса и документов рассчитывается косинусное сходство;
7. N лучших документов и исходный запрос передаются на вход некоторой модели BERT;
8. Результат работы BERT объединяется со всей дополнительной информацией и возвращается в качестве результатов поиска.
9. Заключение

В данной работе был впервые предложен подход к организации системы поиска информации на основе большого ресурса - StackExchange, позволяющий задавать вопросы и получать ответы на естественном языке с дополнительной информацией в виде фрагментов программ. Данный подход легко расширяем на любые веб-ресурсы, форумы, другие языки программирования и использует современные и широко используемые модели машинного обучения.

На базе этого подхода было реализовано программное средство с веб-интерфейсом, позволяющее организовать систему поиска информации по языку Питон.

1. Список литературы
2. Zhaxyhayev D.O., Bakiyev M.N. [Automation algorithm of the information search and fixation in unformalized messages for the management of socially significant intelligent projects //](https://elibrary.ru/item.asp?id=48400415) [Proceedings of Voronezh State University. Series: Systems Analysis and Information Technologies](https://elibrary.ru/contents.asp?id=48400404). – 2022. – [№ 1](https://elibrary.ru/contents.asp?id=48400404&selid=48400415). – С. 139-152.
3. Башлыкова А.А., Растягаев Д.В. Интероперабельность репозиториев версий инструментального программного обеспечения вычислительных комплексов // [Вестник Российского нового университета. Серия: Сложные системы: модели, анализ и управление](https://elibrary.ru/contents.asp?id=45717003). – 2021. – [№ 1](https://elibrary.ru/contents.asp?id=45717003&selid=45717025). – С. 136-142.
4. Ковалев А.Д., Никифоров И.В., Дробинцев П.Д. [Автоматизированный подход к семантическому поиску по программной документации на основе алгоритма DOC2VEC](https://elibrary.ru/item.asp?id=44783233) // [Информационно-управляющие системы](https://elibrary.ru/contents.asp?id=44783228). – 2021. – [№ 1 (110)](https://elibrary.ru/contents.asp?id=44783228&selid=44783233). – С. 17-27.
5. Куликова А.В. Поиск бизнес-информации в электронных информационных системах российских библиотек [//](https://elibrary.ru/item.asp?id=47189347) [Научные и технические библиотеки](https://elibrary.ru/contents.asp?id=47189343). – 2021. – [№ 10](https://elibrary.ru/contents.asp?id=47189343&selid=47189347). – С. 63-80.
6. Митина О.А. Системы поиска информации на основе анализа слабоструктурированного текста // [Национальная Ассоциация Ученых](https://elibrary.ru/contents.asp?id=45599893). – 2021. – [№ 65-2 (65)](https://elibrary.ru/contents.asp?id=45599893&selid=45599902). – С. 30-32.
7. Ронзин В.И., Никитина Е.Ю. Разработка программного модуля поиска нарушений [для интегрированной системы безопасности //](https://elibrary.ru/item.asp?id=43146567) [Вестник Пермского университета. Математика. Механика. Информатика](https://elibrary.ru/contents.asp?id=43146555). – 2020. – [№ 1 (48)](https://elibrary.ru/contents.asp?id=43146555&selid=43146567). – С. 69-7
8. Строева Ю.В. Подход к построению интеллектуального репозитория программных проектов // [Автоматизация процессов управления](https://elibrary.ru/contents.asp?id=49478241). – 2022. – [№ 3 (69)](https://elibrary.ru/contents.asp?id=49478241&selid=49478247). – С. 44-50.
9. Суслов А.А., Садковская Н.Е., Котлярова Н.Ю., Анферова М.С., Кузьмич А.Ю. Разработка алгоритма работы интеллектуальной системы поиска элктрорадио изделий в условиях неполноты информации // [Современные наукоемкие технологии](https://elibrary.ru/contents.asp?id=48061592). – 2022. – [№ 2](https://elibrary.ru/contents.asp?id=48061592&selid=48061627). – С. 130-136.
10. Тихонов Н.А., Будникова И.К. [Организация поиска информации по большим файлам с использованием B-деревьев //](https://elibrary.ru/item.asp?id=44388981) [Информационные технологии в строительных, социальных и экономических системах](https://elibrary.ru/contents.asp?id=44388974). – 2020. – [№ 4 (22)](https://elibrary.ru/contents.asp?id=44388974&selid=44388981). – С. 30-32.
11. Финджоян Д.А., Синица С.Г., Осипян В.О. [Математическая модель и программная архитектура для поиска по зашифрованным данным при подборе персонала //](https://elibrary.ru/item.asp?id=47142699) [Инженерный вестник Дона](https://elibrary.ru/contents.asp?id=47142665). – 2021. – [№ 9 (81)](https://elibrary.ru/contents.asp?id=47142665&selid=47142699). – С. 384-400.
12. Ямашкин С.А., Скворцов М.А., Большакова М.В., Ямашкин А.А. Анализ принципов и методов формирования хранилища временных рядов и организации программных интерфейсов в репозитории нейросетевых моделей // [Современные наукоемкие технологии](https://elibrary.ru/contents.asp?id=46492743). – 2021. – [№ 8](https://elibrary.ru/contents.asp?id=46492743&selid=46492764). – С. 143-147.
13. Ямашкин С.А., Ямашкина Е.О., Ямашкин А.А. Разработка прикладного программного интерфейса для обеспечения обмена данными с репозиторием нейросетефых моделей // [Современные наукоемкие технологии](https://elibrary.ru/contents.asp?id=48617343). – 2022. – [№ 5-2](https://elibrary.ru/contents.asp?id=48617343&selid=48617352). – С. 226-231.