Санкт-Петербургский государственный университет

Кафедра системного программирования

Шумилов Петр Евгеньевич

Разработка системы автоматического разрешения обращений в техническую поддержку

Бакалаврская работа

Научный руководитель:
ст. преп. Я. А. Кириленко

Рецензент:
инженер консультант ООО "САП Лабс" Н. А. Ражев

Санкт-Петербург
2019

SAINT PETERSBURG STATE UNIVERSITY

Software engineering

Petr Shumilov

Development of the incidents resolving system

Graduation Thesis

Scientific supervisor:
senior lecturer I.A. Kirilenko

Reviewer:
Engineer at SAP Labs Nikita Razhev

Saint Petersburg
2019

Оглавление

Введение

На сегодняшний день подавляющее большинство разного рода компаний и организаций имеют систему технической поддержки. Будь это крупный оператор связи или служба доставки цветов, немалую долю в качестве предоставляемого ими продукта играет то, насколько хорошо этот товар или услуга поддерживаются. Ключевым фактором для конечного потребителя является скорость, с которой он может решить те или иные вопросы касаемо предоставляемого ему продукта. С другой стороны перед поставщиком товара или услуги стоит задача как можно быстрее понять, что именно волнует пользователя. Обычно для решения этой проблемы компаниям приходится иметь свой штат сотрудников, готовых как можно быстрее среагировать на проблему пользователя. Затраты на содержание такого отдела могут позволить себе только крупные организации, однако качество такой поддержки намного выше, чем в том случае, когда компания прибегает к услугам аутсорсеров — call-центров.

Хорошей практикой среди организаций, имеющих собственный штат сотрудников поддержки, является разделение этой службы на уровни [1]. Это необходимо для грамотного распределения полномочий между специалистами разной квалификации. Количество уровней определяется в зависимости от вида, размера и потребностей компании. Как правило, типичная структура технической поддержки предполагает наличие не менее, чем двух уровней, первый из которых занимается сбором информации о пользователе, предварительным определением проблемы и предлагает заранее известные решения.

Любое предприятие стремится к снижению издержек в своих бизнес-процессах. Ранее упоминалось, что затраты на содержание штата сотрудников технической поддержки могут являться крупной статьей расходов для организации. Поэтому компании применяют меры по оптимизации затрат на первых уровнях поддержки: уменьшают количество сотрудников, вводят строгие KPI[1] с целью повысить отдачу ра-

[1]Key Performance Indicators — ключевые показатели эффективности

бочей единицы, однако эффективность таких мер спорна [5]. По мнению автора работы, оптимизировать процессы первой линии можно с помощью системы, которая позволит автоматически, на основе текста заявки и некоторых других данных, находить статью в базе готовых решений и предоставлять её пользователю. В случаях, когда готовой инструкции пока нет, система будет выдавать ранжированный список возможных решений, и в самом крайнем случае, будет перенаправлять пользователя к специалисту. При этом на постоянной основе будет запущен процесс, анализирующий разрешённые не автоматически инциденты, и опираясь на эти данные, будет рекомендовать агенту поддержки написать новую статью. Такой подход позволит свести к минимуму нагрузку на первую линию.

Вышеописанная ситуация в индустрии была озвучена сотрудниками петербургского офиса компании SAP [10], что далее повлекло за собой рождение данной работы.

1. Постановка задачи

Целью данной работы является разработка продукта для автоматизированного разрешения обращений в техническую поддержку посредством рекомендации готового решения на основе текста заявки и данных о пользователе. Для достижения данной цели были поставлены следующие задачи:

- провести анализ требований со стороны потенциального заказчика;
- проанализировать существующие программные продукты в данной сфере;
- разработать архитектуру продукта;
- реализовать прототипы следующих сервисов:
  - поиска семантически схожих заявок;
  - поиска наиболее часто встречающихся проблем в истории заявок;
  - определения категории обращения по тексту заявки;
- разработать пользовательский интерфейс;
- провести апробацию прототипа.

2. Требования к системе

В этом разделе описываются требования к системе с обсуждением причин, по которым они были выдвинуты. Требования были разделены на две категории — функциональные и нефункциональные.

2.1. Функциональные требования

2.1.1. Размещение внутри ландшафта компании

В последнее время всё большее количество компаний предпочитают размещать свои бизнес-приложения в облаке или пользоваться облачными сервисами (SaaS, IaaS, PaaS). Безусловным преимуществом такого размещения является отсутствие затрат на содержание оборудования. Однако работа с облачными сервисами подразумевает, что данные компании покидают пределы корпоративное сети. Это ведёт к снижению уровня безопасности некоторых процессов, на что некоторые организации не готовы пойти. Существуют страны, в которых законодательно запрещено передавать и хранить данные пользователей за пределами государства [14]. В то же время, большинство игроков рынка решений для организации технической поддержки, имеющие «умные» сервисы (классификация инцидентов, поиск готовых решений), могут предложить только облачное размещение, в ином случае сильно возрастает стоимость интеграции. В связи с вышеописанной ситуацией было выдвинуто требование о возможности размещения всех сервисов только внутри корпоративной сети.

2.1.2. Интеграция с SAP Solution Manager

В связи с тем, что большинство клиентов SAP и потенциальных заказчиков разрабатываемой системы пользуются продуктом Solution Manager [9], было сформировано требование об интеграции со сценарием обработки инцидентов в Solution Manager.

2.1.3. Интеграция с базами знаний

На сегодняшний день большинство продуктов для обеспечения работы технической поддержки в комплект поставки включают либо модуль либо подпродукт, который занимается менеджментом готовых решений — создание новых статей, дополнение статей, поиск по статьям. Однако многие компании чаще всего уже имеют подобные системы, которые удовлетворяют всем требованиям по функциональности, а также могут предоставлять более широкие возможности. В связи с вышеописанными доводами было внесено требование о возможности интеграции с популярными платформами хранения статей — MediaWiki [8], Confluence [4].

2.1.4. Поддержка взаимодействия через чат-бота

Одним из популярных недостатков многих систем обработки инцидентов являются переусложнённые процедуры создания заявки, а также низкая адаптированность интерфейсов к мобильным платформам. В последнее время индустрия всерьёз начала рассматривать месенджеры, как канал для взаимодействия с пользователем [6]. Это связанно с тем, что разработчики месенджеров активно развивают свои платформы для построения чат-ботов. К тому же, концепция чат-ботов может быть аналогично реализована в рамках веб-интерфейса, который можно легко встроить в корпоративный портал или иные сервисы. В связи со сложившейся конъюнктурой на рынке, было выдвинуто требование о поддержке взаимодействия с системой через чат-бота.

2.2. Нефункциональные требования

2.2.1. Модульность

Как любой налаженный и стабильно работающий процесс система технической поддержки не должна терять возможность обслуживать пользователя даже в случае проведения каких-либо работ или модернизаций. С другой стороны, хорошей практикой внедрения новых возможностей в работающую систему является итеративный подход, при

котором внедряемая система разбивается на компоненты, каждый из которых является обособленным сервисом (модулем) и дорабатывается по отдельности. В связи с вышеописанными доводами, к системе было выдвинуто требование иметь модульную архитектуру.

2.2.2. Скорость внедрения

Так как разрабатываемое решение ориентировано в первую очередь на клиентов, у которых уже есть какой-либо отлаженные процесс поддержки пользователей, было выдвинуто требование, в соответствие которому архитектура продукта должна обеспечить высокую скорость внедрения.

2.2.3. Дружественный интерфейс

Для улучшение пользовательского опыта было выдвинуто требование по созданию легкого и адаптивного интерфейса, способного максимально упростить процесс создания инцидента и дальнейшего взаимодействия с технической поддержкой.

3. Обзор предметной области

За последние десятилетия информационные технологии оказали большое влияние на бизнес-процессы. Повсеместное распространение интернета позволило компаниям быстрее выводить на рынок свои товары и услуги. Параллельно с этим развивались направления связанные с технической поддержкой клиентов, что повлекло за собой появление такого класса систем как сервис-дески (Service Desk). Сервис-деск — это специализированный комплекс, ориентированный на обработку сервисных событий, поступающих в форме обращений пользователей или сообщений систем мониторинга, иными словами — это единая точка контакта между поставщиком услуг и пользователями [1].

На представленной ниже диаграмме описан один из популярных подходов предоставления технической поддержки:

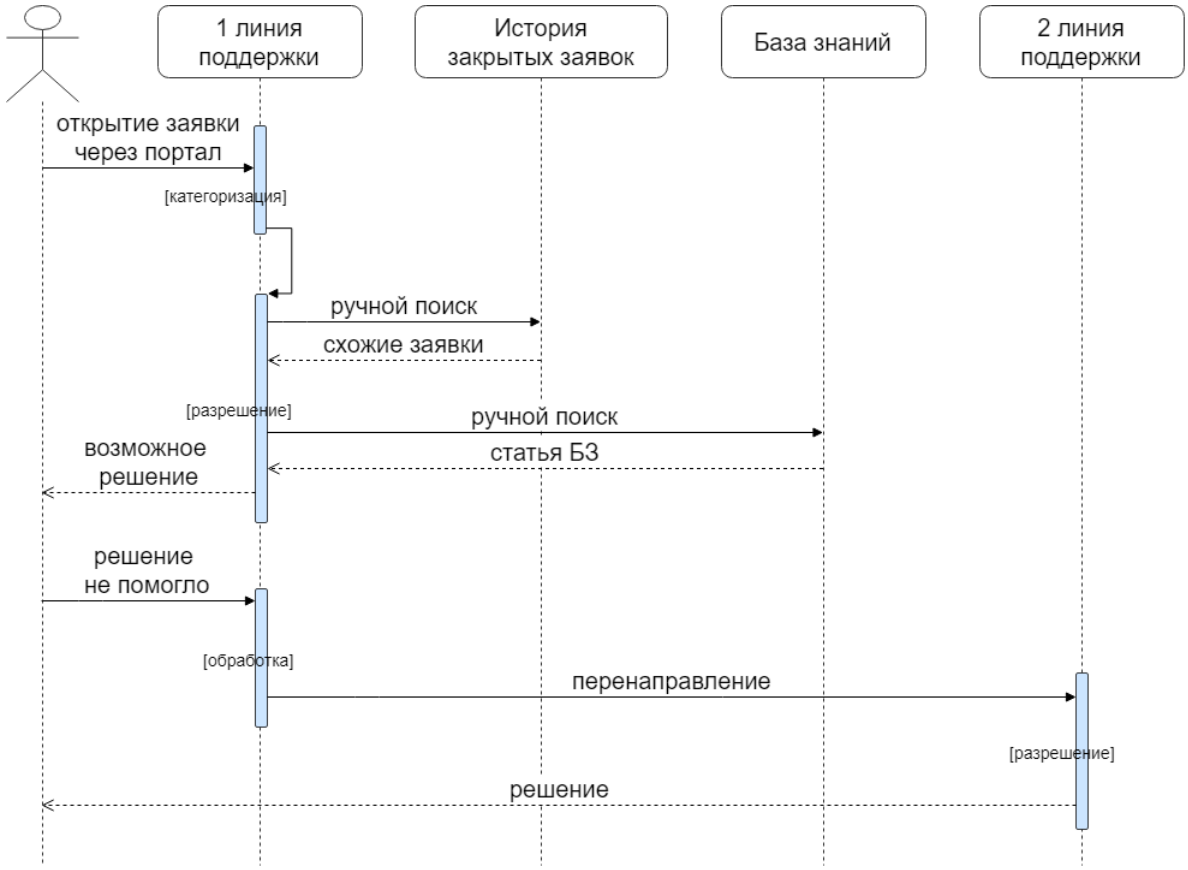

Рис. 1: Популярный процесс

Такой подход подразумевает наличие специализированного храни-

лища готовых решений — базы знаний (knowledge base, БЗ) [13] [7]. Она необходима для структурированного накопления информации о проблемах, возникших у пользователей, и способах их решения. Основная цель — сделать так, чтобы вопросы разрешались до обращения в службу поддержки. К недостаткам подхода, описанного на Рис. 1, можно отнести:

- неудобный доступ — интерфейсы корпоративных порталов чаще всего переусложнены;
- у крупных компаний — тяжёлый процесс определения категории инцидента;
- ручной поиск по статьям базы знаний и истории заявок;
- редкие обновления базы знаний — трудно на основе истории заявок вручную выделить наиболее часто встречающиеся проблемы.

Вышеупомянутые недостатки могут быть нивелированы с помощью внедрения специализированных сервисов, которые способны классифицировать инциденты, искать схожие заявки и выявлять основные топики[2] из истории инцидентов.

3.1. Обзор существующих решений

На сегодняшний день многие производители программного обеспечения имеют в своём арсенале продукт, который занимается менеджментом инцидентов или ITSM[3] в целом. Ситуация обусловлена тем, что существует устойчивый спрос на такие системы со стороны разного рода компаний. Некоторые представители бизнеса имеют свои собственные сервис-дески, которые были разработаны силами внутреннего IT-подразделения. Идёт время и такие системы устаревают — требуют

[2]Топик (тема) — логический субъект предложения, в котором высказывается его смысл.

[3]IT Service Management — подход к управлению и организации IT-услуг, направленный на удовлетворение потребностей бизнеса.

обновления интерфейсов, создания хранилищ готовых решений, внедрение автоматической классификации заявок. Вследствие чего компания встаёт перед выбором: покупка и внедрение абсолютно нового продукта, удовлетворяющего возросшим требованиям, или усовершенствование уже имеющегося. Первый вариант не всегда является целесообразным, так как может спровоцировать высокие издержки, поэтому многие компании останавливаются на втором варианте.

На предмет соответствия выдвинутым ранее требованиям были проанализированы популярные решения на рынке сервис-десков и сервисы для оптимизации работы технической поддержки.

3.1.1. ChangeGear Intelligence Platform

ChangeGear Enterprise Service Desk [12] — это ITSM-решение нацеленное на крупные организации, имеющие, помимо возможности управления инцидентами, модули менеджмента изменений, планирования и контроля сборки программного обеспечения. Данная система поддерживает как SaaS, так и on-premise[4] размещение, также имеет встроенные AI-сервисы[5] для классификации заявок и поиска решений по статьям базы знаний. Концепция продукта подразумевает переход потребителя на экосистему ChangeGear и не предусматривает интеграцию с уже имеющимися сервис-десками.

3.1.2. Intelligent Automation Engine

Продукт от мирового лидера рынка по производству сервис-десков, от компании ServiceNow [11]. Имеет практически такие же возможности, что и ChangeGear, при этом имеет возможность интеграции с SAP Solution Manager и поддерживает платформу Slack в качестве канала для взаимодействия с пользователем. Однако данное решение не может функционировать в условиях полностью изолированного от интернета

[4]On-premise — использование собственных ресурсов предприятия для размещения программного обеспечения.

[5]Artificial Intelligence — свойство компьютерных систем выполнять творческие функции, которые традиционно считаются прерогативой человека.

ландшафта.

3.1.3. SAP Service Ticket Intelligence

SAP Service Ticket Intelligence — продукт, специальной разработанный для построения классификаторов по принципу «текст — категория». Основным преимуществом перед другими продуктами является ориентированность на работу с обращениями в техническую поддержку. Сервис может поставляться только как SaaS.

3.1.4. Выводы

	скорость внедрения	чат-бот интерфейс	on-premise	интеграция с Solman	интеграция с БЗ
ChangeGear	-	+	+	-	-
ServiceNow	-	+	-	+	+
STI	+	-	-	+	+

Таблица 1: Соответствие требованиям

Таким образом, из-за специфических требований, а также других ограничений, ни одно из рассмотренных решений в сфере систем для организации технической поддержки не было взято за основу разрабатываемой системы.

3.2. Обзор технологий

В данном разделе будет проведён обзор основных подходов, применяемых для поиска семантически схожих заявок и выявления основных топиков из истории инцидентов. Далее будут описаны базовые технологии, применяемые для обработки текстов.

3.2.1. Базовый подход к обработке заявки

Базовый рабочий процесс для многих задач, связанных с обработкой текстов выглядит следующим образом: предобработка текста →

представление текста в векторном виде → применение алгоритмов кластеризации, классификации, поиска схожести.

3.2.2. Предобработка текста заявки

Основной задачей предобработки текста является приведение документа на естественном языке к удобному виду для дальнейшей работы. Данный процесс состоит из различных этапов, которые могут отличаться в зависимости от задачи. В качестве примера предобработки можно привести следующие преобразования: удаление всех неважных символов (например, любые символы, не относящиеся к цифро-буквенным), затем токенизация текста — разделение его на индивидуальные слова, далее удаление нерелевантных слов (например, упоминания Google, URL-ы, имена собственные), а также преобразование всех символов в нижний регистр для того, чтобы слова «работа», «Работам» и «РАБОТА» считались одним и тем же токеном. Наконец, можно провести лемматизацию — сведение различных форм одного слова к начальной форме (например, «любовь» вместо «любовью», «любви»).

3.2.3. Представление текста заявки в векторном виде

Векторизация текстов (Word embedding) используется для того, чтобы иметь возможность работать с текстом не как с набором символов, слов и документов, а как с вектором, что позволяет сильно расширить возможности по применению различных алгоритмов.

Один из классических методов отображения текста в вектор заключается в том, что каждому слову, которое встречается в каком-либо документе, сопоставляется измерение в пространстве признаков (словаре). Словарь строится так, что для каждого слова, значение соответствующей координаты будет пропорционально частоте данного слова в документе. Такой подход называется «мешок слов» (Bag of Words). Тексты векторизованные таким методом, можно сравнивать на близость по смыслу, например, с помощью косинусной меры. Однако, такой подход имеет некоторые недостатки. Учет всех встреченных в документах слов

приводит к слишком большой размерности пространства, что в свою очередь может приводить к высокой вычислительной погрешности и низкой скорости работы различных алгоритмов.

Описанный выше подход остаётся применимым в областях, где количество текстов мало и словарь ограничен. Однако существуют другие подходы, основанные на гипотезе, что слова, встречающиеся в одинаковых окружениях, имеют близкую семантику. Одним из таких является Word2Vec [2]. Процесс обучения Word2Vec модели устроен следующим образом: берётся последовательно $2k + 1$ слов, слово в центре является тем, которое должно быть предсказано, а окружающие слова — контекст длины k с каждой стороны. Каждому слову в такой модели сопоставляется уникальный вектор, который меняется в процессе обучения модели. Особенность данной модели является то, что она способна улавливать некоторые семантические свойства слов (Рис. 2).

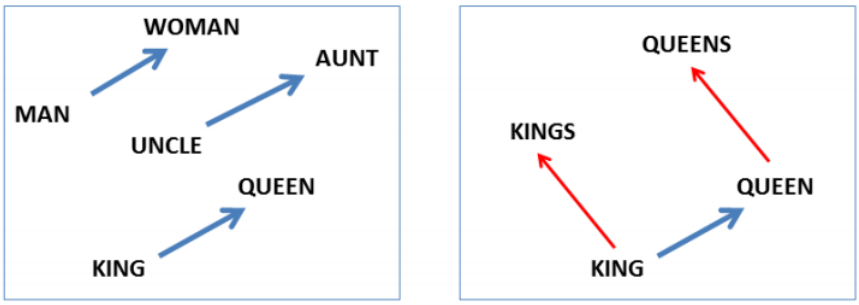

Рис. 2: Семантические свойства Word2Vec

Для повышения качества векторизации документов, существуют методики присваивания веса признаку, позволяющие сфокусироваться на наиболее значимых словах. К таким методам относится TF-IDF (Term Frequency, Inverse Document Frequency). TF-IDF увеличивает вес слова на основании того, насколько редко оно встречается во всех документах, при этом понижает веса тем словам, которые встречаются слишком часто и просто добавляют шум.

4. Архитектура

В данном главе будут представлены архитектурные решения разрабатываемого прототипа и обоснования, почему были использованы те или иные компоненты и подходы.

4.1. Описание процесса

На основании проведённого анализа рынка и с учётом выдвинутых требований был разработан новый процесс (Рис. 3) для организации работы технической поддержки.

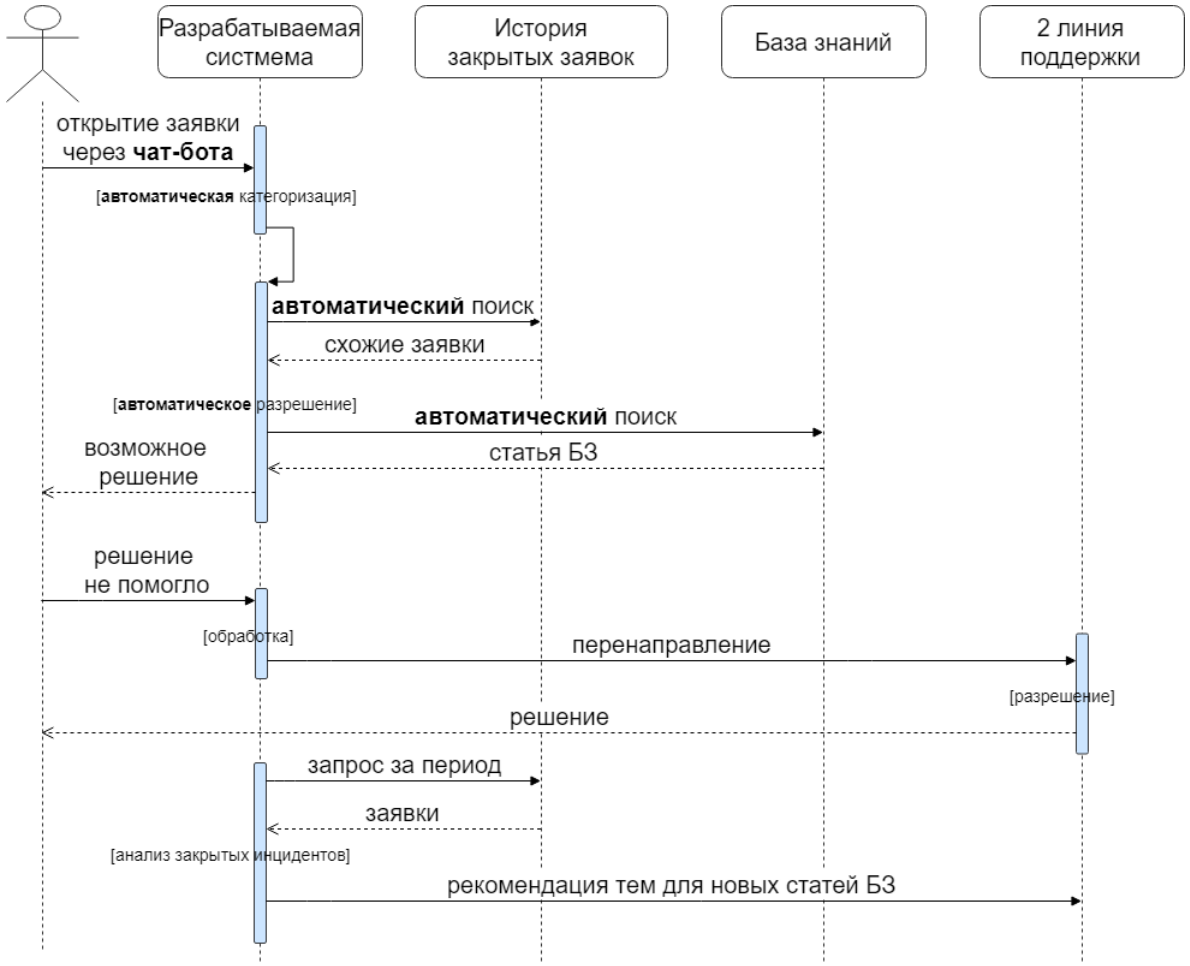

Рис. 3: Разработанный процесс

Основной целью данного подхода является снижение нагрузки на первую линию. Подразумевается, что первоначальное взаимодействие

с пользователем будет полностью возложено на разрабатываемую систему. По сравнению с процессом, представленным на Рис. 1, данный подход имеет следующие отличия:

- открытие заявки происходит через простой и интуитивно понятный интерфейс — через чат-бота;
- определением категории инцидента занимается специальный сервис, а не агент;
- поиск и предоставление возможного решения происходит в автоматическом режиме, что позволяет экономить человеко-часы;
- база знаний поддерживается в актуальном состоянии благодаря специальному процессу, который запускается по истечению некоторого времени и извлекает из истории инцидентов наиболее популярные темы обращений. Далее на их основе высококвалифицированные специалисты описывают способы решения данных проблем.

По мнению автора работы, вышеописанные особенности позволяют улучшить пользовательский опыт и уменьшить количество человека-часов, затрачиваемых на обслуживание одного пользователя.

4.2. Описание компонентов

Для реализации вышеописанного (Рис. 3) процесса, а также с учётом требований была разработана архитектура, высокоуровневое описание которой, представлено на Рис. 4.

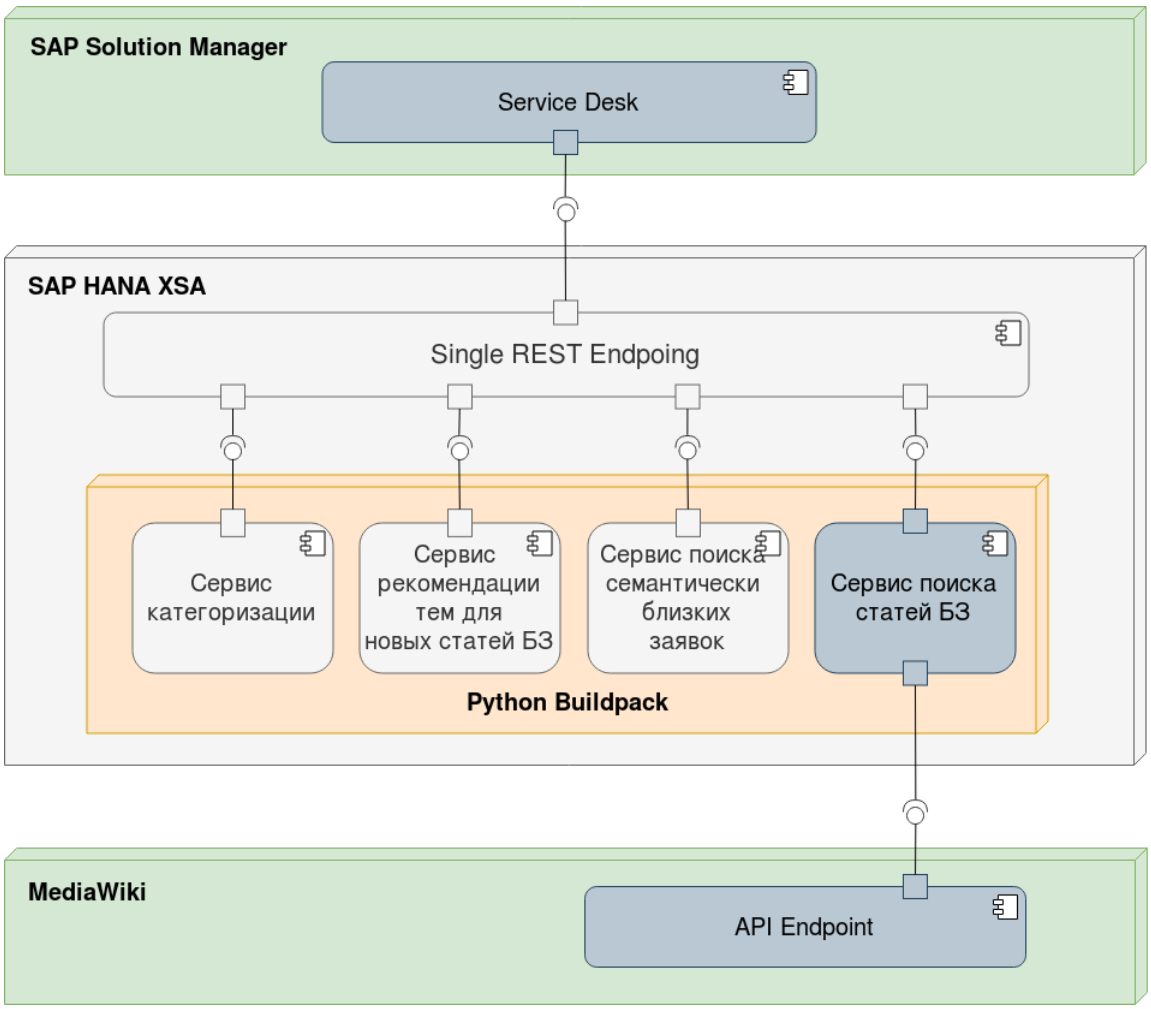

Рис. 4: Архитектура

На Рис. 4 в блоке, имеющем светлым фон, отмечена разрабатываемая система, в блоках с более тёмной заливкой отмечены уже имеющиеся системы заказчика.

Для обеспечения возможности получать отчётность, проводить аудит или эскалировать обращения все инциденты проходят через Solution Manager. Хочется отметить, что некоторые сервис-дески имеют функциональность по добавлению обработчиков на события, возникающие в системе. Например, можно настроить «реакции» на открытие

заявки, на ответ от пользователя и на иные действия. В нашем случае, продукт Solution Manager поддерживает возможность генерации HTTP запроса по открытию инцидента и по изменению его статуса.

Разрабатываемая система состоит из четыре основных компонентов:

- сервис определения категории обращения;
- сервис рекомендации тем для новых статей базы знаний;
- сервис поиска семантически близких заявок;
- сервис поиска статей в базе знаний (реализован не автором данной работы).

Также в системе имеется специальное приложение-прослойка, необходимая для маршрутизации HTTP запросов, исходящих от сервис-деска.

Процесс обработки заявки можно описать следующим образом. После открытия инцидента либо через портал, либо через чат-бота, сервис-деск генерирует HTTP запрос, в теле которого в JSON-формате содержится вся необходимая информация о заявке: заголовок, детальное описание, ссылки на прикреплённые файлы, приоритет и категория (если указаны пользователем). Этот запрос поступает на приложение-прослойку, которое по телу определяет на какой сервис стоит перенаправить данные, преобразовывает их в соответствие с нужным форматом и генерирует HTTP запрос в нужный сервис. По завершению работы сервиса, ответ идёт в обратном направлении, после чего сервис-деск фиксирует возвращённые данные (например, категорию) и эмитирует событие о смене статуса заявки, которое генерирует новый запрос. Последовательность повторяется до этапа предоставления возможного решения (см. Рис. 3).

4.2.1. Сервис определения категории

Основной задачей данного компонента является определение категории заявки на основе её текста и заголовка.

Для корректной работы сервиса требуется обучить классификатор, которому необходимы данные вида «заголовок, описание заявки — категория». Чаще всего у компаний есть обширная история разрешённых инцидентов с правильно определённой категорией. Данный сервис способен получать выборку заявок посредством oData-запроса в Solution Manager или напрямую из реляционной базы данных.

Сервис имеет два основных метода:

1. POST /train — получает на вход временной интервал, по которому нужно взять историю и произвести обучение;

2. POST /categorize — получает на вход заголовок и текст заявки, возвращает категорию.

В качестве языка программирования для реализации данного компонента был выбран Python из-за возможность простого прототипирования, гибкости, огромного количества инструментов, позволяющих работать с данными и различными алгоритмами машинного обучения. Для реализации REST [6] API был выбран фреймворк Flask, так как он является простым в использовании и идеально подходить для создания «веб-обёрток» над методами.

Чтобы повысить качество классификации, сервис предобрабатывает данные, на которых будет происходить обучение. Данный процесс включает в себя:

- токенизацию — разделения текста на предложения и слова;

- удаление нерелевантных символов и стоп слов — знаки препинания, предлоги, некоторые частицы;

- исправление грамматических ошибок;

- раскрытие аббревиатур;

- поиск и удаление имён собственных;

[6]REST — архитектурный стиль взаимодействия компонентов распределённого приложения в сети.

- нормализацию, в данном случае лемматизацию;
- векторизацию.

Вышеописанный методы, были применены с целью уменьшения количества шума и улучшения показателей категоризации. В качестве метода классификации была выбрана модель логистической регрессии, так как она продемонстрировала наилучшие показатели на данных клиента (91000 заявок, 100 классов) по метрики accuracy, которая является наиболее интуитивно понятной для представителей бизнеса.

4.2.2. Сервис рекомендации тем для новых статей

Данный сервис предназначен для предоставления списка наиболее часто встречающихся проблем в виде перечня сгруппированных тэгов, сформированных на основании истории инцидентов. Другими словами, если из 4000 обращений за два месяца, 2000 были о проблемах с доступом к системе X, 500 о сложностях с открытием отпуска, а 1500 о невозможности сгенерировать счёт-фактуру, то сервис сформирует три топика с описаниями вида: «доступ, проблема с доступом, прошу открыть, SMD», «заявление, отпуск, создать, не могу» и «счёт-фактура, не могу, сч, создать». Данная информация, полученная из истории заявок за последнее время, позволит специалистам технической поддержки своевременно создать новую статью для базы знаний и сильно снизить количество инцидентов, поднявшихся выше первого уровня.

Сервис имеет один основной метод: GET /topics — получает на вход временной интервал, по которому нужно взять историю и сформировать списки наиболее часто встречающихся проблем.

Выбор языка программирования, фреймворков и подходов к реализации предобработки истории аналогичны сервису определения категории. В качестве метода построения топиков были опробованы идеи, описанные в статье «Comparing different term weighting schemas for Topic Modeling» [3]. Ниже в таблице представлены результаты сравнения двух методов по метрике Adjusted Rand Index и субъективной метрике понятности топика для человека.

Подход / Метрика	ARI	Удовлт. пользователя	Коэф. силуэта	Пример топика
Word2Vec + K-Means + LDA	0.66	+	0.20	«изменить сокращенное наименование кредитора»
BoW + LDA	0.74	-	отсут	«создавать дебитор согласно приложение заявка»

Таблица 2: Сравнение подходов

Тестирование проводилось на исторических данных, состоящих из 91000 заявок и содержащих дополнительно пару виды «заявка — статья из базы знаний, решившая проблему». Исходя из того, что пользоваться сервисом будут специалисты технической поддержки, было принято решение использовать первый вариант, так субъективно он выдаёт более понятные наборы тэгов.

4.2.3. Сервис поиска семантически близких заявок

Основная задача данного компонента состоит в поиске по тексту заявки семантически близких инцидентов в истории обращений и выдаче их в ранжированном по схожести порядке. Например, если пользователь обращается с проблемой «не удаётся выполнить вход на портал, необходимы сертификаты», и ранее были разрешены заявки «не могу войти в админ-панель», «авторизация на портале», то сервис вернёт сначала идентификатор второй заявки, а затем первой.

Сервис имеет один основной метод: GET /search – получает на вход текст заявки и какое количество наиболее близких по смыслу заявок необходимо вернуть, другими словами «Top N».

Выбор языка программирования, фреймворков и подходов к реализации предобработки истории аналогичны сервису определения категории. В качестве меры схожести между двумя заявками было выбрано косинусное расстояние между их векторами. Векторизация заявки происходит с помощью суммирование векторов признаков каждого слова

с учётом меры TF-IDF. Числовое представление слова извлекается из ранее обученной на 91000 заявок Word2Vec модели.

Для обоснования применения косинусного расстояния был предложен метод, позволяющий произвести оценку того, насколько использованная мера уместна. Будем считать, что заявки «сильно схожи», если у них одна категория и они закрыты с помощью одной статьи базы знаний, «схожи», если они закрыты только с помощью одной статьи базы знаний, «слабо схожи», если у них одна категория и «разные», в остальных случаях. Затем, для некоторого большого (относительно размера истории обращений) числа заявок найдём «Top N» близких заявок и подсчитаем сколько в каждой выдаче «сильно схожих», «схожих», «слабо схожих» и «разных». Таким образом можно интуитивно понятно для представителей бизнеса показать, в скольких процентах случаев сервис находит «сильно схожие» заявки.

4.2.4. Чат-бот

С учетом требований к прототипу была разработана подсистема для взаимодействия с пользователем посредством командного чат-бота. На Рис. 5 представлена её высокоуровневая архитектура (границы системы имеют ярко красную заливку).

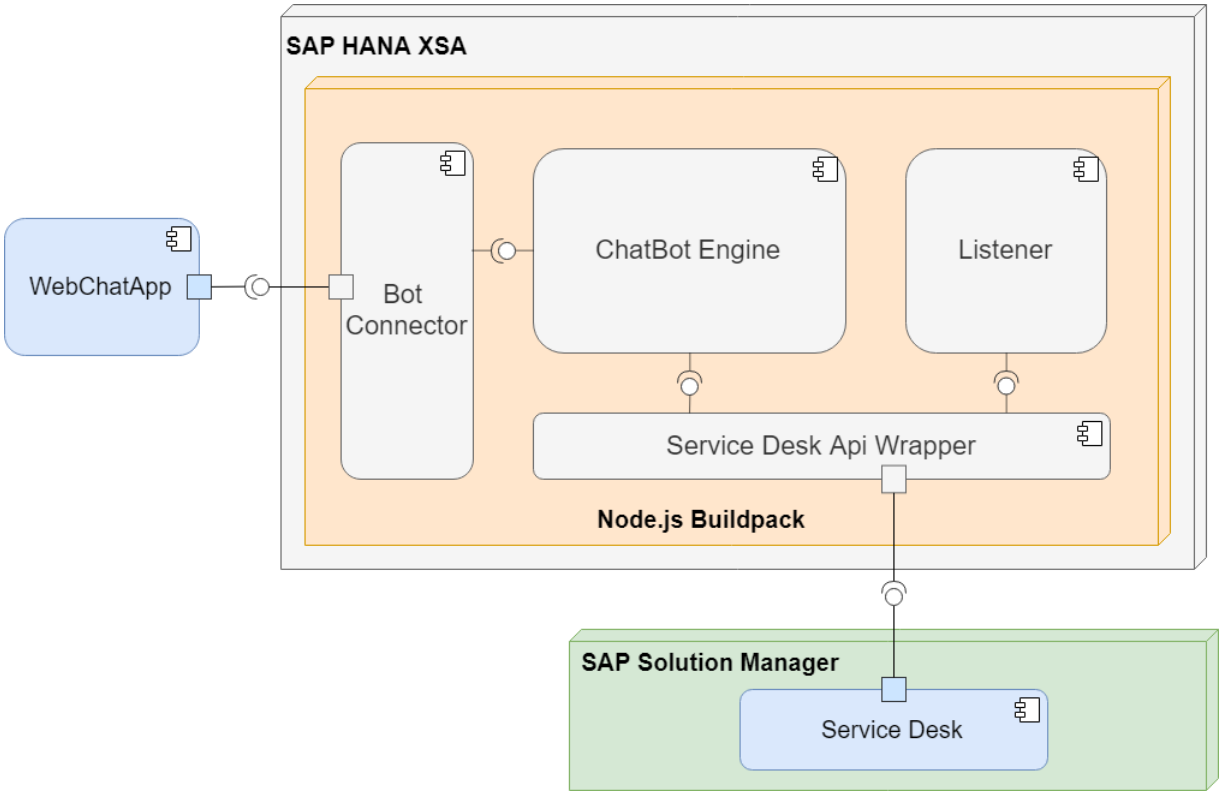

Рис. 5: Архитектура чат-бот платформы

Система состоит из три основных компонентов: бот-коннектора (Bot Connector), обёртки над API сервис-десков (Service Desk API Wrapper) и модуля реализующего поведение бота (ChatBot Engine). В свою очередь, элемент диаграммы под названием Listener необходим в том случае, когда сервис-деск не имеет возможности генерировать HTTP запросы по каким-либо событиям, например, в случае добавление нового сообщения агентом поддержки в инцидент.

Компонент Bot Connector необходим для отображения сущностей (сообщений, картинок, стикеров) конкретного мессенджера или вебчата в абстракции, которыми далее будет удобно оперировать в компоненте, реализующим поведение бота.

Модуль Service Desk API Wrapper является прослойкой, схожей по концепции с бот-коннектором, т.е. решает задачу отображения сущностей конкретного сервис-деска в сущности бизнес логики.

Компонент ChatBot Engine является главной частью вышепредставленной архитектуры (Рис. 5) и реализует поведение бота, а также объединяет все остальные модули.

Для описания поведения бота был разработан специальный декларативный язык (Рис. 6), оперирующий тремя основными понятиями: intent — это команда (например, «/reset»), по которой вызывается skill. Skill — это сущность, описывающая какое-либо «умение» бота, например, «умение» создавать инцидент или сбрасывать пароль в какой-либо системе. Skill выспрашивает данные у пользователя, необходимые для завершения «умения» (например, в какой системе нужно сбросить пароль) и аккумулирует их в себе. Когда все необходимые данные собраны, skill вызывает action — действие, которое, например, выполняет запрос к сервис-деску и открывает инцидент.

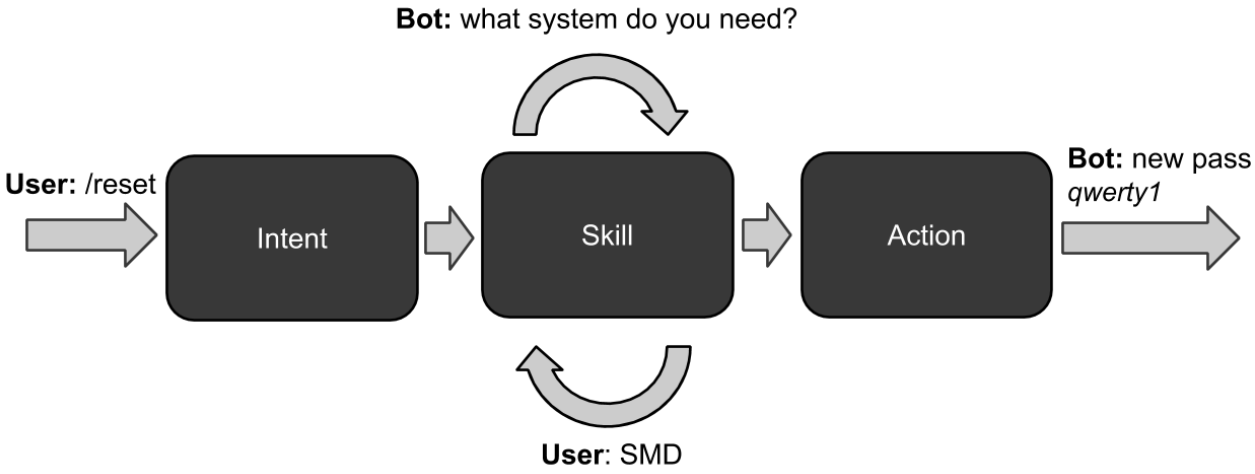

Рис. 6: Описание поведения бота

5. Апробация прототипа

Разрабатываемый прототип был доставлен одному из клиентов компании SAP (обособленное подразделение ООО «САП Лабс» в Санкт-Петербурге) и на момент написания данного текста находится в стадии тестирования. Со стороны группы представителей технической поддержки компании-клиента, участвующих в тестировании прототипа, была отмечена простота внедрения сервисов, а также высокая информативность тем, рекомендуемых сервисом для анализа истории заявок.

Заключение

В ходе выполнения данной выпускной квалификационной работы были достигнуты следующие результаты:

- проведён анализ требований со стороны потенциального заказчика и со стороны разработчика;
- проанализированы существующие решения от крупных IT-компаний (ServiceNow, SunWeb Software) и выявлены их особенности;
- с учётом требований разработана модульная архитектура продукта;
- реализован прототип системы, включающий следующие компоненты:
  - сервис поиска семантически схожих заявок;
  - сервис поиска наиболее часто встречающихся проблем в истории заявок;
  - сервис определения названия продукта по тексту заявки;
- разработан пользовательский интерфейс, позволяющий взаимодействовать с системой посредством чат-бота;
- прототип системы был опробован клиентом компании SAP и получил положительные отзывы пользователей.

Список литературы

[1] Foundation ITIL. ITIL Open Guide. — Access mode: https://www.itlibrary.org/.

[2] Mikolov Tomáš. Word2Vec. — 2013. — Access mode: https://en.wikipedia.org/wiki/Word2vec.

[3] Truică Ciprian-Octavian. Comparing different term weighting schemas for Topic Modeling. — 2015. — Access mode: https://www.academia.edu/30880388/Comparing_different_term_weighting_schemas_for_Topic_Modeling.

[4] atlassian.com. Confluence is an open and shared workspace. — 2019. — Access mode: https://www.atlassian.com/software/confluence.

[5] habr.com. KPI, или пособие по командному самоубийству. — 2018. — online; accessed: https://habr.com/post/152445/.

[6] itsm.tools. In ITSM, Where Do the Good Ideas Come From? — 2019. — Access mode: https://itsm.tools/2017/01/05/itsm-influencers-2/.

[7] joetheitguy.com. 10 Ways Your IT Service Desk Will Benefit from a Knowledge Base. — 2019. — Access mode: https://www.joetheitguy.com/2018/05/23/10-ways-your-it-service-desk-will-benefit-from-a-knowledge-base

[8] mediawiki.org. MediaWiki is a collaboration and documentation platform brought to you by a vibrant community. — 2019. — Access mode: https://www.mediawiki.org/wiki/MediaWiki.

[9] sap.com. SAP Solution Manager. — 2018. — Access mode: https://www.sap.com/products/solution-manager.html.

[10] sap.com. About SAP. — 2019. — Access mode: https://www.sap.com/corporate/en.html.

[11] servicenow.com. Intelligent Automation Engin. — 2019. — Access mode: https://www.servicenow.com/products-by-category.html.

[12] sunviewsoftware.com. IT Solutions for Your Modern Digital Workplace. — 2019. — Access mode: https://www.sunviewsoftware.com/products.

[13] wikipedia.org. Knowledge base. — 2019. — Access mode: https://en.wikipedia.org/wiki/Knowledge_base.

[14] Яровая Ирина Анатольевна. Пакет Яровой. — 2016. — Access mode: http://www.consultant.ru/cons/cgi/online.cgi?req=doc&base=LAW&n=201078&dst=1000000001.