УДК 004.89
DOI: 10.22213/2410-9304-2024-3-103-112

Предобработка данных для модели Doc2Vec при автоматической классификации текстовых запросов в службе технической поддержки провайдеров программного обеспечения

О. М. Шаталова, доктор экономических наук, ИжГТУ имени М. Т. Калашникова, Ижевск, Россия
А. А. Кузьмин, аспирант, ИжГТУ имени М. Т. Калашникова

Ведущие компании-провайдеры программного обеспечения (ПО), как правило, реализуют функции технической поддержки клиентов; это становится важным фактором продвижения и конкурентоспособности продуктов и услуг провайдеров ПО на глобальных рынках. Высокий поток запросов и их неоднородность – функциональная, временная, языковая и др. – определяют значимость эффективной классификации, которая, в свою очередь, позволяет оптимально распределить этот поток между специалистами службы технической поддержки (СТП) и/или автоматизировать обработку заявок с использованием сложившейся базы знаний. Определение класса запросов является слабо формализованной задачей. Для компаний, накопивших значительный объем данных о запросах клиентов, становится возможной автоматизация классификации с помощью методов машинного обучения и моделей обработки естественного языка, таких как, например, Word2Vec, FastText, BERT, GPT и др. Принято считать, что эффективность классификации в этом случае зависит главным образом от применяемой модели. Однако на качество модели существенное влияние оказывает характер данных, применяемых для ее обучения. Обзор литературы показал высокий исследовательский интерес к методам автоматической классификации текстовых запросов, специфицированным к условиям деятельности СТП провайдеров ПО. Вместе с тем нужно отметить слабое внимание к вопросам о влиянии стадии предобработки данных на качество моделей, разрабатываемых в этой предметной области. Цель статьи состояла в уточнении содержания техник предобработки и анализе их влияния на эффективность классификации текстовых запросов с учетом специфики деятельности СТП провайдеров ПО. В ходе исследования было изучено содержание стадий процесса автоматической классификации текстовых заявок с учетом особенностей данных (клиентских текстовых запросов); сформирован релевантный комплекс специфицированных методических и инструментальных средств; проведена их экспериментальная апробация на основе открытых данных одного из глобальных провайдеров ПО (DevExpress). Для апробации была использована база данных, включающая 165 000 текстовых запросов, подготовленных необходимым образом. Результаты исследования показали, что предварительная обработка может улучшить метрики классификации – F-мера, точность (Precision), полнота (Recall) – с 77 до 79 %, а также позволяет значительно сократить размерность текстовых данных (на 48,2 %) и повысить скорость обучения модели (на 26,5 %) без потери в точности, что обеспечивает экономичность и оперативность в использовании вычислительных ресурсов.

Ключевые слова: обработка естественного языка, предобработка данных, Doc2Vec, классификация текстов, заявки технической поддержки.

Введение

В настоящее время многие крупные провайдеры ПО реализуют функции технической поддержки клиентов по вопросам использования поставляемых продуктов или услуг. Эффективность реализации таких функций существенно влияет на репутацию компании и уровень удовлетворенности потребителей.

Для снижения трудоемкости и повышения качества обработки текстовых запросов клиентов, накопления базы знаний центров технической поддержки провайдеров ПО возрастает значимость задачи классификации поступающих текстовых запросов пользователей. Решение этой задачи может быть автоматизировано. Но нужно исходить из того, что текстовые заявки представляют собой неструктурированные данные, а семантика запроса трудно улавливается стандартными компьютерными алгоритмами. Для решения такого класса задач используются методы обработки естественного языка (Natural Language Processing, NLP) [1]. Благодаря NLP-технологиям возможно выделять ценную информацию из текстовых заявок клиентов, в том числе для решения задачи их классификации.

Процедуры NLP часто разбиваются на несколько стадий: сбор данных, предобработка данных, векторизация и обучение классификатора [2]. Наибольший исследовательский интерес связан со стадиями векторизации и обучения классификатора. Так, например, в [3] рассматривались различные стратегии векторизации текстов заявок и изучалось влияние функций отображения документов в векторное пространство на эффективность классификации, производимой моделью BERT и ее модификациями. В [4] было изучено влияние подбора классификатора и был проведен сравнительный анализ классических методов и ансамблевых методов для категоризации текстовых заявок пользователей (так называемых тикетов, от англ. tickets – билет, карточка), поступающих в СТП. В [5] изучаются архитектуры нейронных сетей и стандартные методы машинного обучения для автоматической категоризации тикетов. По итогам анализа литературных источников было выявлено, что влияние стадии предварительной обработки данных на эффективность классификации изучается намного реже. Вместе с тем в последние годы наблюдается тренд на исследование влияния стадии предобработки текстов. Например, влияние предобработки на качество текстовой классификации изучено в [6] и [7]; в частности, рассмотрено влияние техник предобработки и их комбинаций на точность (accuracy) моделей в разных предметных областях. В [8] изучено влияние отдельных стадий предобработки на классификацию медицинских показаний. В [9] составлен обширный набор

техник предобработки, в том числе довольно редких, и проведен анализ влияния каждой из них на качество предсказаний нейросетевых моделей типа «трансформер». Таким образом, в последнее время уделяется внимание стадии предобработки, однако не было найдено публикаций, представляющих исследование методов предобработки для классификации текстовых запросов, специфических для деятельности СТП провайдера ПО. Хотя специфика таких заявок и ее влияние на текстовую классификацию были представлены в статье LyubinetsV. etal. [10], авторами этой статьи был изучен только аспект выделения именованных сущностей, связанных с продукцией компании, вместе с тем в этой работе были оставлены без рассмотрения другие особенности запросов СТП, а также вопросы реализации стадии предобработки.

Цель представленного в статье исследования состояла в уточнении содержания техник предобработки и анализе их влияния на эффективность классификации текстовых запросов с учетом специфики деятельности СТП провайдеров ПО. Поставленная цель раскрывается следующим набором задач исследования: 1) описание процесса автоматической классификации текстовых заявок с учетом особенностей данных (клиентских текстовых запросов СТП провайдеров ПО); 2) формирование комплекса специфицированных методических и инструментальных средств для реализации процедуры автоматической классификации текстовых заявок (для формирования базы данных, векторизации текстовых документов, решения задачи классификации, оценки эффективности полученных решений); 3) экспериментальная оценка влияния техник предобработки данных на результаты автоматической классификации текстовых запросов, проведенная на основе открытых данных одного из глобальных провайдеров ПО (DevExpress).

Используемые подходы и методы

Состав методов, использованных в проведенном исследовании, структурирован в соответствии с содержанием процедуры автоматической классификации текстовых запросов; ее структурное представление приведено на рис. 1.

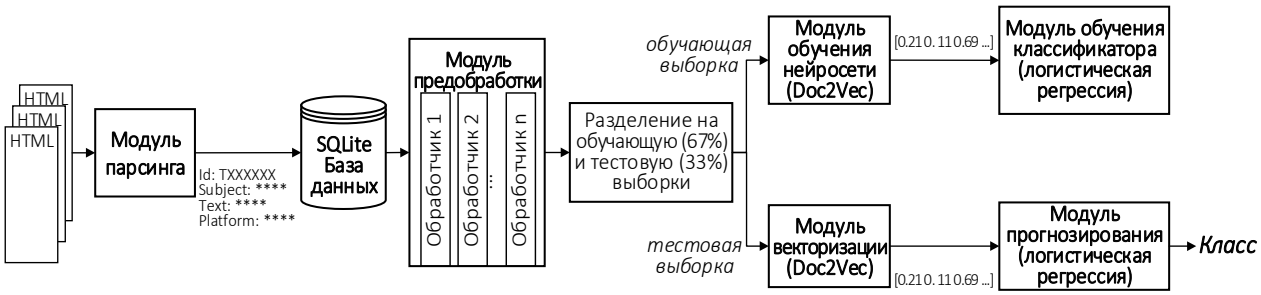

Рис. 1. Алгоритм процедуры автоматической классификации текстовых запросов СТП

Fig. 1. Algorithm for the Automated Classification of Tickets in Support Centers

Первый этап состоит в формировании базы данных, включающей текстовые запросы, сложившиеся за всю историю взаимодействия с пользователями. На этом этапе востребованы методы сбора данных с внешнего источника, автоматизированного синтаксического анализа (так называемого парсинга, от англ. parsing – анализ) данных и их структуризации, организации хранения данных.

Второй этап – предобработка данных – имеет своей целью очистку данных от шума и снижение размерности элементов базы данных (поле «текст запроса»). Для такой предобработки задействованы как стандартные методы, так и методы, кастомизированные к специфике текстовых запросов.

На третьем этапе реализуется векторизация текстовых документов, т. е. процесс преобразования текста в числовой формат, что позволяет представить документы в формате, который используется моделями машинного обучения [11]. Векторное представление документа также позволяет хранить информацию в численном виде, в том числе семантическую информацию. Эта закодированная семантическая информация позволяет улучшить качество классификаторов. Для векторизации в рамках исследования была задействована нейросетевая модель Doc2Vec, обученная на сформированной обучающей выборке.

Четвертый этап состоит, собственно, в классификации текстовых заявок; для этого была задействована модель логистической регрессии, обученная с использованием векторизованных заявок, сформированных на третьем этапе.

Методы формирования базы данных

Поскольку для исследования предполагается использовать открытые данные провайдера ПО, размещаемые на веб-ресурсе, необходимо было решить задачу автоматизированного извлечения данных со страниц веб-ресурсов (web scraping). Формирование соответствующих запросов с веб-сервера (GET-запросов) для сбора необработанных HTML-страниц заявок проводилось с использованием метода get класса Client Session из библиотеки Asyncio/Aiohttp. Структуризация данных, направленная на создание полей: «тема запроса», «текст запроса», «метка продуктовой линейки», реализована на основе автоматизированной процедуры преобразования текстовых данных в необходимых структурированных форматах. С этой целью был использован синтаксический анализатор (так называемый парсер, от англ. parse – анализировать), написанный на языке Python. Структура синтаксического анализатора (парсера) включает 4 модуля:

модуль 1 – web scraping – извлечение HTML-страниц, соответствующих публичным заявкам;

модуль 2 – построение структурированных данных из неструктурированного текстового формата; были выделены участки полученных в предыдущем модуле HTML-страниц, которые содержат полезную информацию о заявке, в JSON-файлы;

модуль 3 – перевод структуры данных в сформированных JSON-файлах в битовую последовательность (так называемая сериализация) для создания необходимых объектов песред внесением в базу данных (БД);

модуль 4 – загрузка результата сериализации в БД. Модуль загрузки отправляет запросы на вставку записей в базу данных с помощью метода connect библиотеки aiosqlite. Для подключения к БД метод connect требует указания строки подключения, идентифицирующей базу данных (путь к файлу с БД). При вставке записей в БД копируется информация из полей объектов, полученных в результате сериализации (номер запроса, тема запроса, название платформы, текст запроса). Для организации хранения данных задействована файловая СУБД, реализованная средствами библиотеки SQLite.

Предобработка текстов заявок

Каждая заявка представляет собой личную переписку на английском языке с участием двух и более лиц на некоторую тему, в которой, как правило, содержатся вопрос клиента и ответ на вопрос сотрудника технической поддержки. Суммарный текст одной заявки содержит в среднем 305 слов, т. е. их можно считать длинными текстами.

Процедуре предобработки предшествует первичный анализ данных – идентификация базовых лингвистических единиц, подлежащих обработке (так называемая токенизация, от англ. token – признак, опознавательный знак и т. п.); выявление наиболее частотных базовых лингвистических единиц (токенов), в том числе ключевых терминов, именованных сущностей, связанных со спецификой работы провайдера ПО; и др. Для идентификации базовых лингвистических единиц была использована библиотека Nltk (метод word_tokenize).

Процесс токенизации может быть представлен математически следующим образом: пусть S – это строка как последовательность из n символов $c_i : S = \{c_i\}_{(i=1)}^{n}$; тогда токенизация может быть представлена функцией T, которая сопоставляет последовательность S и последовательность токенов: $T : S \to \{t_j\}_{j=1}^{m}$, где t_j – это j-й токен, а m – это число токенов, сгенерированных в результате применения функции T; в данной работе использовалась процедура разбиения на токены по символу пробела.

Обработка полученных результатов для получения частотного распределения токенов проведена средствами библиотеки Nltk – метод plot класса FreqDist (ключевые термины в ходе первичного анализа данных не определяются).

Определение частоты токенов может быть представлено в следующем виде: пусть D – это набор данных, состоящий из k строк: $D = \{S_l\}_{l=1}^{k}$, где S_l – это l-я строка в наборе данных; тогда частотой токена t_j является функция $f(t_j)$: $f(t_j) = \sum_{l=1}^{k} \sum_{j=1}^{m_l} \delta(t_j, t_{jl})$, где m_l – количество токенов в l-й строке, а $\delta(t_j, t_{jl})$ – это функция Кронекера $\delta(t_j, t_{jl}) = \begin{cases} 1, t_j = t_{jl} \\ 0, t_j \neq t_{jl} \end{cases}$; тогда нужно найти множество токенов $\{t\}$ такое, что $f(t_1) \geq f(t_2) \geq \ldots \geq f(t_k) \geq f(t) \forall t \in U : \{t_1 \cdots t_k\}$, где U – это множество уникальных токенов во всем наборе данных.

Для предобработки были задействованы как стандартные методы, так и дополнительные обработчики, которые учитывают сложившуюся специфику текстов переписки клиентов с агентами СТП, а также позволяют исключить фрагменты текста, которые не несут семантической нагрузки и могут быть отнесены к категории шума.

Стандартные процедуры предобработки включают [12]:

– удаление знаков пунктуации: $R(S) = (c_i | \delta(c_i) = 1 \forall i \in \{1 \ldots n\})$, где $\delta(c) = \begin{cases} 1, c \in P \\ 0, c \notin P \end{cases}$, где P – множество знаков пунктуации;

– удаление стоп-слов (не несущих уникального семантического смыслаили слова общей лексики): $RSW(U) = (t_j | \delta(t_j) = 1 \forall j \in \{1 \ldots m\})$, где U – это множество токенов, $\delta(t_j) = \begin{cases} 1, t_j \in SW \\ 0, t_j \notin SW \end{cases}$, а SW – множество токенов, представляющих стоп-слова;

– приведение к нижнему регистру $L(S) = (lower(c_i) | c_i \in S)$, где $lower$ – функция сопоставления ASCII (от англ. American standard code for information interchange – стандарт кодирования знаков латинского алфавита, цифр, некоторых специальных знаков и управляющих последовательностей, принятый в качестве основного способа представления текстовых данных для машинной обработки), символу другого ASCII символа, представляющего строчной вариант буквы;

– автоматическую морфологическую разметку (частеречную разметку или POS-Tagging): $POS(t) = \underset{ps1, ps2 \cdots psn}{argmax} P(ps_1, ps_2 \ldots ps_n | t_1 \ldots t_m)$, где ps – множество всех возможных частей речей, а P – условная вероятность принадлежности токена к одной из частей речи;

– нормализацию, включая: а) поиск основы слова и отсечение его от окончаний, префиксов и суффиксов (так называемый стемминг, от англ. stemming – находить происхождение): $STEM(S) = (stem(t_j) | t_j \in S)$, где $stem$ – функция отображения токена в токен на основе заранее заданных правил (в работе использовался PorterStemmer); б) приведение словоформы к ее пер-

воначальной словарной форме – лемме (так называемая лемматизация, от англ. lemmatization): $LEMMA(S) = \left(lemma\left(t_j, POS\left(t_i\right)\right) | t_j \in S\right)$, где $LEMMA(S)$ – это функция отображения из токена в токен на основе части речи токена, заранее заданных правил и словарей.

Для реализации стандартных техник предобработки были использованы: метод word_tokenize, метод stopwords.words, метод stem класса PorterStemmer из библиотеки Nltk (токенизация, удаление неспецифичных стоп-слов, стэмминг соответственно); метод load (модель «en_core_web_sm») для загрузки модели из библиотеки Spacy (частеречная разметка, лемматизация).

Кастомизированные процедуры предобработки зависят от результатов первичного анализа данных, который выявляет специфику тестов. Экспериментальные результаты такого анализа и соответствующие методы кастомизированной предобработки приведены во второй части статьи.

Методы векторизации

Преобразование текстового документа в числовой вектор, который содержит информацию о семантике и контексте документа, позволяет передать в модель машинного обучения информацию о текстах в виде данных в числовом формате. Эффективность классификации текстовых запросов в значительной мере зависит от того, насколько числовые векторы сохраняют синтаксическую и семантическую информацию, поэтому простое кодирование каждого слова в предложении может быть недостаточным и становится необходимым использование релевантных моделей векторизации. Так, распространение получили модели векторизации, основанные на дистрибутивной семантике, то есть на определении смысла текстовой единицы по окружающим ее другим текстовым единицам. Одной из таких моделей является модель Doc2Vec, которая и была использована в данной работе для векторизации текстов. Механизм работы модели Doc2Vec принципиально схож с механизмом работы модели векторизации слов Word2Vec [13]. Word2Vec – это модель, которая обучается таким образом, что в результате обучения на больших массивах лингвистических данных каждому строковому представлению слова сопоставляется его векторное представление, сохраняющее смысл слова. При обучении модель Word2Vec запоминает контекст слова, то есть какие слова находились вблизи данного в обучающей выборке. Обученная модель Word2Vec реализуется двумя методами: 1) CBoW (от англ. Continuous Bag of Words – сплошной мешок слов) – позволяет предсказать пропущенное слово по контексту; 2) Skip-gram (от англ. Skip – пропускать, grammar – грамматика) – позволяет предсказать наиболее вероятное окружение для целевого слова [14].

Модель Doc2Vec позволяет получить векторное представление отдельных предложений и абзацев [15]. Для этого могут быть использованы два метода, объединенных в группу PV (от англ. Paragraph Vectors – векторы абзацев): PV-DM (от англ. Distributed Memory – распределенная память) и PV-DBoW (от англ. Distributed Bag of Words – распределенный мешок слов). Содержание этих методов состоит в следующем [16]: метод PV-DM позволяет «предсказывать вектор следующего слова в документе» на основе нейросети, «обученной по вектору документа и некоторой последовательности векторов слов»; в методе PV-DBoW «нейросеть обучается предсказывать все слова в документе по его вектору». Векторы документов, полученные с помощью модели Doc2Vec, могут быть применены в таких задачах, как кластеризация документов, поиск близких по смыслу документов и классификации [17].

Обучение модели Doc2Vec проводилось с помощью размеченной обучающей выборки, которая была сформирована случайным образом из составленной БД: для каждого набора обработчиков в модуле предварительной обработки проведено 10 параллельных экспериментов, в каждом из которых использовались разные выборки; результаты получившихся показателей усреднены. Размер обучающей выборки был принят на уровне 67 % от всего объема данных. Реализация Doc2Vec проведена с помощью Python библиотеки Gensim (методы build_vocab, train и infer_vector класса Doc2Vec). В данной работе был использован следующий набор параметров модели Doc2Vec: размер векторов 500, размер окна для прогнозирования слов 15, минимальная частотаобнаруженияслова в выборке должна составлять 2. Количество эпох при обучении составило 20.

Методы классификации

На этапе классификации каждый вектор заявки сопоставляется с меткой каждой продуктовой линейки. Для классификации был реализован вероятностный подход к анализу данных и использован метод логистической регрессии; логистическая регрессия возвращает вероятность принадлежности вектора каждому из изучаемых классов. Данный метод выбран, поскольку он продемонстрировал свою эффективность в задачах классификации при использовании больших объемов данных [18]. Для реализации логистической регрессии была использована библиотека Scikit-Learn с базовыми гиперпараметрами (методы fit и predict класса Logistic Regression). Методы класса Logistic Regression принимают на вход только числовые параметры, поэтому необходима предварительная векторизация текстов.

Методы оценки качества классификации

Для оценки качества классификации, а также для влияния стадии предобработки на результат классификации были использованы общепринятые в машинном обучении метрики Precision (точность), Recall (полнота), F-мера [19]. Данные метрики основаны на следующих показателях результата: истинно положительные (True Positive, TP) – количество объектов, которые модель верно предсказала как принадлежащие к положительному классу; истинно отрицательные (True Negative, TN) – количество объектов, которые модель верно предсказала как принадлежащие к отрицательному классу; ложно положительные (FalsePositive, FP) – количество объектов, которые модель неправильно предсказала как принадлежащие к положительному

классу (ошибки первого рода); Ложно отрицательные (FalseNegative, FN) – количество объектов, которые модель неправильно предсказала как принадлежащие к отрицательному классу (ошибки второго рода).

Метрика Precision характеризует, насколько модель правильно классифицирует объекты как положительные классы: $Precision = \frac{TP}{TP + FP}$. Метрика Recall измеряет способность модели обнаруживать все положительные случаи: $Recall = \frac{TP}{TP + FN}$. F-мера представляет собой сбалансированную метрику, которая учитывает как Precision, так и Recall, и определяется как среднее гармоническое между Precision и Recall: $F = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$.

• *Средства программной реализации методов решения задачи*

• *классификации текстовых заявок*

Программная реализация включала использование публичных библиотек, которые представлены в табл. 1.

Таблица 1. **Состав библиотек, использованных в исследовании задачи классификации текстовых запросов**

Table 1. **Libraries Used for theTicket Classification Task**

Названия библиотек	Назначение библиотек	Использование библиотек в данной работе
Numpy	Проведение математических операций с инструментами линейной алгебры, такими как многомерные массивы, матрицы, векторы	Операции с векторами и матрицами
Pandas	Проведение анализа данных	Статистический анализ базы заявок, очистка данных, хранение данных в памяти
Scikit-learn	Модуль для машинного обучения	Использование реализации логистической регрессии, разделение выборки на обучающую и тестовую, расчет метрик классификации
Nltk (Natural Language Toolkit)	Статистическая обработка естественного языка для английского языка	Удаление стоп-слов, стэмминг, токенизация
Spacy	Продвинутые техники обработки естественного языка	Частеречная разметка, распознавание именованных сущностей, лемматизация
VADER(Valence Aware Dictionary and sEntiment Reasoner)	Инструмент для анализа тональности текстов	Удаление эмоциональных слов
Gensim	Инструменты для тематического моделирования, информационного поиска и других задача обработки естественного языка	Использование реализации Doc2Vec
Asyncio/Aiohttp	Асинхронное написание кода и асинхронные HTTP клиент и сервер	Формирование асинхронных запросов на сервер провайдера ПО DevExpress для сбора HTML-страниц
Beautifulsoup4	Извлечение данных из HTML и XML файлов	Извлечение полезных данных из HTML-страниц, содержащих тексты заявок
SQLite	Хранение информации в БД	Хранение данных о заявках

Источник: составлено авторами.

Результаты исследования: влияние предобработки данных на результаты автоматической классификации текстовых запросов СТП провайдера ПО

Формирование базы данных текстовых заявок СТП для их классификации

Информационную основу исследования составили открытые данные компании DevExpress о запросах пользователей. Эти данные размещены на веб-ресурсе «Support Center Knowledge Base» («База знаний Центра поддержки») (SCKB) в формате HTML-страниц; каждая HTML-страница соответствует одной заявке и содержит переписку, состоящую из нескольких постов от клиента и от агентов технической поддержки. Каждый запрос, как правило, связан с информацией относительно функциональности одного или более компонентов какой-либо продуктовой линейки компании DevExpress. В результате сбора данных с ресурса SCKB было собрано более 750 000 заявок.

В соответствии с алгоритмом классификации тикетов (рис. 1), на этапе сбора данных был реализован модуль парсинга. Это позволило сохранить собранные заявки (тикеты) в таблице базы данных SQLite, содержащей соответствующие поля: идентификатор заявки (INTEGER), тема заявки (TEXT), текст постов (TEXT), название платформы (TEXT).

Анализ собранных и структурированных данных показал следующее:

а) незначительная часть записей (0,17 %) не содержит сведений о продуктовой линейке; эти данные были удалены из базы;

б) другая часть записей идентифицируется с *определенной* продуктовой линейкой DevExpress, общее количество которых составляет 38 ед.;

в) структура запросов в разрезе продуктовых линеек является очень неоднородной (рис. 2), при этом в качестве достаточно представительных можно рассматривать данные только по 11 продуктовым линейкам – ASP.NETWebForms, WinForms, AppFrameworks (UI, API, ORM), HTMLJS, WPF, Reporting, BusinessIntelligence, VCL, ASP.NETMVC, ASP.NETCore, Angular.

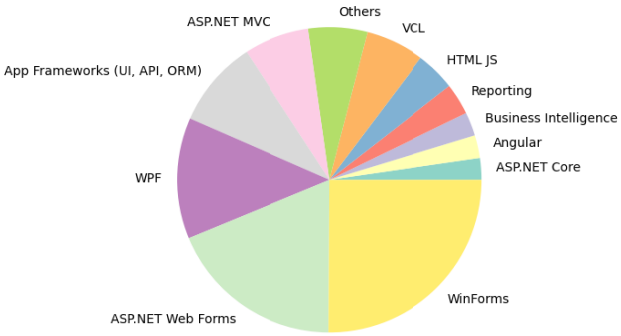

Рис. 2. Распределение заявок по продуктовым линейкам в сформированной базе заявок (исходный вариант)

Fig. 2. Distribution of tickets by platforms in the gathered dataset (raw data)

Количество строк БД было сбалансировано таким образом, чтобы каждая продуктовая линейка была представлена одинаковым количеством запросов; в итоге, размер БД сократился до 150000 объектов.

Предобработка, векторизация, классификация текстовых заявок СТП

На стадии предобработки был проведен предварительный анализ текстов заявок, который позволил выделить их особенности, а также стоп-слова и специфические шумы, которые оказывают значительное влияние на качество классификации.

Для выявления стоп-слов тексты заявок разделили на токены, на основе которых построен словарь из 1080055 единиц. Наиболее частотные токены представлены на рис. 3.

Как видно из графика, наиболее частотные токены содержат небуквенные символы и стоп-слова (неспецифические), некоторые из них представлены однобуквенными словами. Большая часть таких слов не несет смысловой нагрузки для определения класса заявки. Это свидетельствует об актуальности, в первую очередь, стандартных процедур предобработки.

К категории шума были отнесены такие фрагменты текстов, как фразы-шаблоны (например, “i look for ward to hearing from you.”, “you are welcome. Should you have further questions, feel free to ask.”), имена собственные (клиентов и агентов технической поддержки), слова вежливости, эмоционально окрашенные слова, опечатки (“gird” вместо “grid”, “autopatically” вместо “automatically”), ссылки на другие страницы, представленные в виде URL-строк.

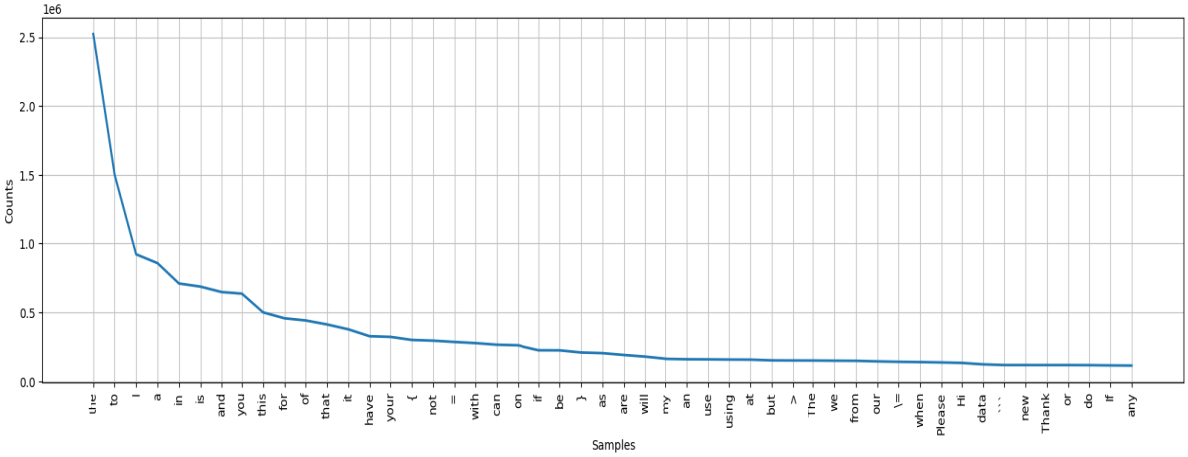

Рис. 3. Частота обнаружения токенов в базе заявок (до использования техник предобработки)

Fig. 3. Unigram frequency distribution in the dataset (before data preprocessin

Специфические условия текстов состоят в следующем: обилие специальной для предметной области лексики и аббревиатур (VB, AI, IP, ORM, control, ticket, button); наличие именованных сущностей, характерных для продукции/услуг компании (ASPxGridView, DxButton); включение в текст заявки участков кода; некоторые участки кода обособлены тройными кавычками в разметке страницы, а некоторые встроены в текст; также существенным условием становится наличие спецсимволов, определяемых синтаксисом языков программирования, – подчеркивание, дефис, точка для введения методов и свойств классов; упоминание используемых продуктов и связей между ними, а также описание, какая именно функциональность работает некорректно и в каких условиях; упоминание в ответах отдельных функций, приводящих к решению проблемы, и характер их использования.

На основании проведенного анализа специфики текстов был сформирован ряд дополнительных условий предобработки: 1) удаление однобуквенных слов, 2) удаление адресов электронной почты, 3) удаление имен собственных за исключением наименований компонентов, продуктов и других имен,

способствующих классификации заявки, 4) удаление участков кода, 5) удаление адресов сайтов (URL-строк), 6) удаление эмоциональных слов.

Удаление однобуквенных слов производилось без использования сторонних библиотек. Удаление адресов электронной почты, URL-строк, а также участков кода производилось с помощью регулярных выражений. Удаление имен собственных проводилось с помощью библиотеки Spacy (метод pipe класса spacy.lang.en.English и поле ents класса Doc), которая позволяет выделить именованные сущности и их категории (имя человека, топоним и т. д.). Удаление эмоциональных слов производилось с помощью словаря из библиотеки Vader (метод make_lex_dict класса Sentiment Intensity Analyzer).

Обработчики тестировались отдельно друг от друга, то есть на этапе предварительной обработки использовался только один обработчик для очистки от шумов. Отдельно рассматривался случай, при котором очистка от шумов производилась в несколько этапов, где текст последовательно подвергался обработке с помощью всех ранее упомянутых техник. Для этого сценария использовалась следующая последовательность обработчиков: приведение к нижнему регистру, удаление адресов электронной почты, удаление URL-строк, удаление участков кода, удаление небуквенных символов, токенизация, частеречная разметка, лемматизация, удаление однобуквенных слов, удаление стоп-слов, удаление эмоциональных слов, удаление имен собственных.

Очищенный набор данных далее был векторизирован с помощью модели Doc2Vec. В результате применения модели каждая запись набора данных представляла числовой вектор размера 500 и метку платформы. Набор данных, состоящий из числовых векторов и меток классов, использовался моделью логистической регрессии в модуле классификации. Далее обученная модель логистической регрессии была использована для прогнозирования классов для векторизованной тестовой выборки.

Оценка результатов классификации текстовых заявок СТП

При сравнении результатов реальных и спрогнозированных значений меток классов были посчитаны метрики качества классификации (по п. 1.5). Результаты классификации (в форме основных метрик качества), полученные при каждом из рассмотренных способов предобработки данных, представлены в табл. 2.

Помимо метрик качества классификации, нами были изучены параметры использования вычислительных ресурсов: 1) размер файла обучающей и тестовой выборки датасета, используемой для обучения модели Doc2Vec и файла для логистической регрессии; 2) среднее время обучения модели Doc2Vec. Значения этих параметров также приведены в табл. 2.

Таблица 2. **Показатели эффективности классификации текстовых заявок, достигаемые при различных методах предобработки данных**

Table 2. **Ticket classification metrics achieved for different data preprocessing methods**

Используемые обработчики		Точность	Полнота	F-мера	Размер выборки (обучающей и тестовой), МБ	Ср. время обучения Doc2Vec, мин.
Текст без предварительной обработки		0,7729	0,7761	0,7745	414,9	49
Кастомизированные	Удаление эмоциональных слов	0,7713	0,7757	0,7735	351,6	44
	Удаление адресов электронной почты	0,7745	0,7745	0,7745	398,8	46
	Удаление имен собственных	0,7772	0,7768	0,777	412,4	46
	Удаление ссылок	0,7699	0,7719	0,7709	366,7	44
	Удаление участков кода	0,7259	0,8309	0,7749	398,5	46
	Удаление однобуквенных слов	0,7812	0,7812	0,7812	309,5	44
Стандартные	Удаление небуквенных символов	0,8136	0,8188	0,8162	354,9	44
	Лемматизация с частеречной разметкой	0,7841	0,7841	0,7841	367,4	47
	Лемматизация без частеречной разметки	0,7751	0,7822	0,7786	313,6	44
	Удаление неспецифичных стоп-слов	0,7724	0,7724	0,7724	251,0	39
	Стэмминг	0,7736	0,7736	0,7736	300,3	44
Все стадии (нормализация: лемматизация с частеречной разметкой)		0,7952	0,7952	0,7952	214,7	36

Источник: составлено авторами.

Представленные в табл. 2 результаты свидетельствуют, что в отсутствие процедуры предобработки данных достигается 77%-я точность классификации. На повышение качества классификации в наибольшей мере повлияло удаление небуквенных символов; возможно, это обусловлено уменьшением шума при сохранении языковой ясности в векторах документов, созданных Doc2Vec. Эта техника улучшила способность модели улавливать семантические связи и контекстную информацию, что привело к повышенной точности классификации и F-меры. Некоторые этапы предобработки, такие как удаление участков кода,

влияют на метрики по-разному, притом что метод позволил увеличить полноту (до 0,8309), это произошло за счет уменьшения точности (0,7259). При последовательном применении всех указанных техник предобработки качество модели Doc2Vec улучшается, однако метрики не являются оптимальными; это объяснимо тем, что предобработка связана с удалением большого количества важной для классификации информации, в том числе информации, содержащейся в участках кода и именованных сущностях. Вместе с тем метод предобработки, включающий весь набор обработчиков, показал свою наибольшую эффективность по совокупности параметров: наряду с повышением качества классификации (с 77 до 79 %), метод позволяет значительно сократить размерность текстовых данных (на 48,2 %) и скорость обучения модели (на 26,5 %) без потери в точности. При использовании более сложных нейросетей со множеством данных о весах нейронов, сокращение времени обучения является важной задачей.

Выводы

Технологии NLP, как правило, включают стадию предобработки данных, реализуемую как стандартными универсальными, так и специфицированными методами, учитывающими особенности предметной области автоматизации.

В исследовании возможностей автоматической классификации текстовых запросов в деятельности СТП провайдеров ПО, реализуемой с помощью модели Doc2Vec, экспериментально была проверена гипотеза о влиянии стадии предобработки данных на ее эффективность. При этом были реализованы сбор данных с внешнего источника, парсинг данных и их структуризация. Предобработка проводилась с учетом специфики текстовых запросов, а также специфики деятельности СТП провайдера ПО. Выполненный анализ позволил выявить существенные специализированные правила предобработки данных в реализации процедуры автоматической классификации запросов. Действенность этих правил была проверена при реализации модели Doc2Vec (для векторизации текстовых запросов) и реализации, собственно, процедуры классификации методом логистической регрессии.

Результаты исследования показали значимость как универсальных, так и кастомизированных техник предобработки, поскольку только метод, включающий полный комплекс обработчиков, показал свою наибольшую эффективность, т. к. позволил повысить качество классификации и при этом значительно сократить размерность текстовых данных (на 48,2 %) и повысить скорость обучения модели (на 26,5 %), что обеспечивает экономичность и оперативность в использовании вычислительных ресурсов.

Библиографические ссылки

1. Sun W, Cai Z, Li Y, Liu F, Fang S, Wang G. Data Processing and Text Mining Technologies on Electronic Medical Records: A Review. J Healthc Eng. 2018 Apr 8;2018:4302425. doi: 10.1155/2018/4302425. PMID: 29849998; PMCID: PMC5911323.

2. May M.C., Neidhöfer J., Körner T., Schäfer L., Lanza G. Applying Natural Language Processing in Manufacturing, Procedia CIRP, Vol. 115, 2022, pp. 184-189, ISSN 2212-8271, https://doi.org/10.1016/j.procir.2022.10.071.

3. Zangari A. Ticket automation: An insight into current research with applications to multi-level classification scenarios / A. Zangari, M. Marcuzzo, M. Schiavinato // Expert Systems with Applications. – 2023. – Т. 225. – С. 119984. DOI https://doi.org/10.1016/j.eswa.2023.119984.

4. Paramesh S. P. Classifying the Unstructured IT Service Desk Tickets Using Ensemble of Classifiers / S. P. Paramesh, C. Ramya, K. S. Shreedhara // 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS). 2018. P. 221-227. DOI 10.1109/CSITSS.2018.8768734.

5. Han J. Towards Effective Extraction and Linking of Software Mentions from User-Generated Support Tickets / J. Han, K. H. Goh, A. Sun, M. Akbari // Proceedings of the 27th ACM International Conference on Information and Knowledge Management : CIKM '18. – New York, NY, USA : Association for Computing Machinery, 2018. P. 2263-2271. DOI 10.1145/3269206.3272026.

6. Uysal A. K. The impact of preprocessing on text classification / A. K. Uysal, S. Gunal // Information Processing & Management. 2014. Vol. 50. No. 1. Pp. 104-112. DOI https://doi.org/10.1016/j.ipm.2013.08.006.

7. Etaiwi W. The Impact of applying Different Preprocessing Steps on Review Spam Detection / W. Etaiwi, G. Naymat // Procedia Computer Science. 2017. Vol. 113. Pp. 273-279. DOI https://doi.org/10.1016/j.procs.2017.08.368.

8. Kashina M. Preprocessing of unstructured medical data: the impact of each preprocessing stage on classification / M. Kashina, I. D. Lenivtceva, G. D. Kopanitsa // Procedia Computer Science. 2020. Vol. 178. Pp. 284-290. DOI https://doi.org/10.1016/j.procs.2020.11.030.

9. Siino M. Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers / M. Siino, I. Tinnirello, M. La Cascia // Information Systems. 2024. Vol. 121. Pp. 102342. DOI https://doi.org/10.1016/j.is.2023.102342.

10. Lyubinets V. Automated Labeling of Bugs and Tickets Using Attention-Based Mechanisms in Recurrent Neural Networks / V. Lyubinets, T. Boiko, D. Nicholas // 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP). 2018. Pp. 271-275. DOI 10.1109/DSMP.2018.8478511.

11. Dvoynikova A. Analytical review of approaches for tonality recognition of Russian text data Аналитический обзор подходов к распознаванию тональности русскоязычных текстовых данных / A. Dvoynikova, A. Karpov // Information and Control Systems. 2020. Pp. 20-30. DOI 10.31799/1684-8853-2020-4-20-30.

12. Frank E. Data preprocessing techniques for NLP in BI / E. Frank, J. Oluwaseyi, G. Olaoye. 2024.

13. Agrawal R. Developing bug severity prediction models using word2vec / R. Agrawal, R. Goyal // International Journal of Cognitive Computing in Engineering. – 2021. Vol. 2. Pp. 104-115. DOI https://doi.org/10.1016/j.ijcce.2021.08.001.

14. Jaya Hidayat T. H. Sentiment analysis of twitter data related to Rinca Island development using Doc2Vec and SVM and logistic regression as classifier / T. H. Jaya Hidayat, Y. Ruldeviyani, A. R. Aditama и др. // Procedia Computer Science. 2022. Vol. 197. Pp. 660-667. DOIhttps://doi.org/10.1016/j.procs.2021.12.187.

15. *Ковалев А. Д., Никифоров И. В., Дробинцев П. Д.* Автоматизированный подход к семантическому поиску по программной документации на основе алгоритма

Doc2Vec // Информационно-управляющие системы. 2021. № 1. С. 17–27. Doi:10.31799/1684-8853-2021-1-17-27.

16. *Пархоменко П. А., Григорьев А. А., Астраханцев Н. А.* Обзор и экспериментальное сравнение методов кластеризации текстов // Труды ИСП РАН. 2017. Т. 29, вып. 2. С. 161–200. DOI: 10.15514/ISPRAS-2017-29(2)-6.

17. *Ковалев А. Д., Никифоров И. В., Дробинцев П. Д.* Автоматизированный подход к обнаружению семантически близких запросов заказчика в системе отслеживания ошибок Jira // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и Технические Науки. 2021. № 05/2. С. 61–67. DOI 10.37882/2223-2966.2021.05-2.15.

18. Maalouf M. Weighted logistic regression for large-scale imbalanced and rare events data / M. Maalouf, M. Siddiqi // Knowledge-Based Systems. 2014. Vol. 59. Pp. 142-148. DOI https://doi.org/10.1016/j.knosys.2014.01.012.

19. Fränti P. Soft precision and recall / P. Fränti, R. Mariescu-Istodor // Pattern Recognition Letters. 2023. Vol. 167. Pp. 115-121.

References

1. Sun W, Cai Z, Li Y, Liu F, Fang S, Wang G. Data Processing and Text Mining Technologies on Electronic Medical Records: A Review. J Healthc Eng. 2018 Apr 8;2018:4302425. doi: 10.1155/2018/4302425. PMID: 29849998; PMCID: PMC5911323.

2. May M.C., Neidhöfer J., Körner T., Schäfer L., Lanza G. Applying Natural Language Processing in Manufacturing,Procedia CIRP, Vol. 115, 2022, pp. 184-189, ISSN 2212-8271, https://doi.org/10.1016/j.procir.2022.10.071.

3. A. Zangari, M. Marcuzzo, M. Schiavinato, A. Gasparetto, and A. Albarelli, “Ticket automation: An insight into current research with applications to multi-level classification scenarios,” Expert Syst. Appl., vol. 225, p. 119984, 2023, doi: https://doi.org/10.1016/j.eswa.2023.119984.

4. S. P. Paramesh, C. Ramya, and K. S. Shreedhara, “Classifying the Unstructured IT Service Desk Tickets Using Ensemble of Classifiers,” in 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS), 2018, pp. 221–227. doi: 10.1109/CSITSS.2018.8768734.

5. J. Han, K. H. Goh, A. Sun, and M. Akbari, “Towards Effective Extraction and Linking of Software Mentions from User-Generated Support Tickets,” in Proceedings of the 27th ACM International Conference on Information and Knowledge Management, in CIKM ’18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 2263–2271. doi: 10.1145/3269206.3272026.

6. A. K. Uysal and S. Gunal, “The impact of preprocessing on text classification,” Inf. Process. Manag., vol. 50, no. 1, pp. 104–112, 2014, doi: https://doi.org/10.1016/j.ipm.2013.08.006.

7. W. Etaiwi and G. Naymat, “The Impact of applying Different Preprocessing Steps on Review Spam Detection,” Procedia Comput. Sci., vol. 113, pp. 273–279, 2017, doi: https://doi.org/10.1016/j.procs.2017.08.368.

8. M. Kashina, I. D. Lenivtceva, and G. D. Kopanitsa, “Preprocessing of unstructured medical data: the impact of each preprocessing stage on classification,” Procedia Comput. Sci., vol. 178, pp. 284–290, 2020, doi:https://doi.org/10.1016/j.procs.2020.11.030.

9. M. Siino, I. Tinnirello, and M. La Cascia, “Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers,” Inf. Syst., vol. 121, p. 102342, 2024, doi: https://doi.org/10.1016/j.is.2023.102342.

10. V. Lyubinets, T. Boiko, and D. Nicholas, “Automated Labeling of Bugs and Tickets Using Attention-Based Mechanisms in Recurrent Neural Networks,” in 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), 2018, pp. 271–275. doi: 10.1109/DSMP.2018.8478511.

11. A. Dvoynikova and A. Karpov, “Analytical review of approaches for tonality recognition of Russian text dataAnaliticheskii obzor podkhodov k raspoznavaniyu tonal'nosti russkoyazychnykh tekstovykh dannykh,” Inf. Control Syst., pp. 20–30, Aug. 2020, doi: 10.31799/1684-8853-2020-4-20-30. (in Russ.).

12. E. Frank, J. Oluwaseyi, and G. Olaoye, “Data preprocessing techniques for NLP in BI,” Apr. 2024.

13. R. Agrawal and R. Goyal, “Developing bug severity prediction models using word2vec,” Int. J. Cogn. Comput. Eng., vol. 2, pp. 104–115, 2021, doi: https://doi.org/10.1016/j.ijcce.2021.08.001.

14. T. H. Jaya Hidayat, Y. Ruldeviyani, A. R. Aditama, G. R. Madya, A. W. Nugraha, and M. W. Adisaputra, “Sentiment analysis of twitter data related to Rinca Island development using Doc2Vec and SVM and logistic regression as classifier,” Procedia Comput. Sci., vol. 197, pp. 660–667, 2022, doi: https://doi.org/10.1016/j.procs.2021.12.187.

15. Kovalev A.D., Nikiforov I.V., Drobintsev P.D. *Avtomatizirovannyj podhod k semanticheskomu poisku po programmnoj dokumentacii na osnove algoritma Doc2Vec* [Automated approach to semantic search through software documentation based on Doc2Vec algorithm]. *Informatsionno-upravliaiushchie sistemy*, 2021, no. 1, pp. 17-27 (in Russ.). DOI:10.31799/1684-8853-2021-1-17-27.

16. Parhomenko P.A., Grigor'ev A.A, Astrahancev N.A. *Obzor i eksperimental'noe sravnenie metodov klasterizacii tekstov* [Review and experimental comparison of text clustering methods]. *Trudy ISP RAN.* Vol. 29, no. 2, 2017, pp. 161-200 (in Russ.). DOI: 10.15514/ISPRAS-2017-29(2)-6.

17. Kovalev A., Nikiforov I., Drobintsev P. *Avtomatizirovannyj podhod k obnaruzheniyu semanticheski blizkih zaprosov zakazchika v sisteme otslezhivaniya oshibok Jira*[An automated approach to finding semantically related customer requests in the Jira bug tracking system]. *Sovremennaya nauka: aktual'nye problemy teorii i praktiki. Seriya: Estestvennye i Tekhnicheskie Nauki.* 2021, no. 05/2, pp. 61-67 (in Russ.). DOI 10.37882/ 2223-2966.2021.05-2.15.

18. M. Maalouf and M. Siddiqi, “Weighted logistic regression for large-scale imbalanced and rare events data,” Knowl.-Based Syst., vol. 59, pp. 142–148, 2014, doi: https://doi.org/10.1016/j.knosys.2014.01.012.

19. P. Fränti and R. Mariescu-Istodor, “Soft precision and recall,” Pattern Recognit. Lett., vol. 167, pp. 115–121, 2023, doi: https://doi.org/10.1016/j.patrec.2023.02.005.

Preprocessing Text Data for the Doc2Vec Model in the Automatic Classification of Tickets in Support Centers of Software Providers

O. M. Shatalova, DSc in Economics, Kalashnikov Izhevsk State Technical University, Izhevsk, Russia
A. A. Kuzmin, Post-graduate, Kalashnikov Izhevsk State Technical University, Izhevsk, Russia

Leading software providers typically implement customer technical support functions, which are crucial for promoting and enhancing the competitiveness of their products and services in global markets. The high volume and heterogeneity of support tickets (functional, temporal, linguistic, etc.) highlight the importance of efficient classification systems. Effective classification optimizes the distribution of these tickets among support center specialists and automates their processing using an established knowledge base. However, classifying these tickets is a loosely formalized task. For companies that have accumulated substantial data on customer requests, automating classification through machine learning methods and natural language processing models, such as Word2Vec, FastText, BERT, and GPT, becomes feasible. It is generally accepted that classification effectiveness primarily depends on the model employed. Nevertheless, the quality of these models is significantly influenced by the nature of the training data. Literature review of the reveals significant research interest in methods for the automatic classification of tickets specifically tailored to the operational conditions of software provider support centers. However, there is a noticeable gap in the literature regarding the impact of data preprocessing on the quality of these models. The article aims to clarify the techniques of data preprocessing and analyze their impact on the effectiveness of text classification, considering the specificity of software provider support centers. This study examines the stages of the automatic classification process for tickets, accounting for the unique characteristics of the data (customer text requests). A relevant set of specified methodological and instrumental tools was developed and tested using open data from a global software provider (DevExpress). The testing involved a database of 165,000 tickets. The study's results indicate that preprocessing can improve classification metrics such as F-measure, Precision, and Recall from 77% to 79%. Additionally, preprocessing significantly reduces the dimensionality of text data (by 48.2%) and increases model training speed (by 26.5%) without loss of accuracy, ensuring cost-efficiency and operational efficiency in the use of computational resources.

Keywords*:* natural language processing, data preprocessing, Doc2Vec, text classification, support tickets.

Получено: 12.06.24

Образец цитирования

Шаталова О. М., *Кузьмин А. А.* Предобработка данных для модели Doc2Vec при автоматической классификации текстовых запросов в службе технической поддержки провайдеров программного обеспечения // Интеллектуальные системы в производстве. 2024. Т. 22, № 3. С. 103–112. DOI: 10.22213/2410-9304-2024-3-103-112.

For Citation

Shatalova O.M., Kuzmin A.A. [Preprocessing text data for the Doc2Vec model in the automatic classification of tickets in support centers of software providers]. *Intellektual'nye sistemy v proizvodstve.* 2024, vol. 22, no. 3, pp. 103-112. DOI: 10.22213/2410-9304-2024-3-103-112.