ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ»

На правах рукописи



Алексеева Дарья Сергеевна

МОДЕЛЬ, АЛГОРИТМ И ПРОГРАММНОЕ СРЕДСТВО КВАЛИФИКАЦИИ НЕСТРУКТУРИРОВАННЫХ ДАННЫХ В АВТОМАТИЗИРОВАННЫХ СИСТЕМАХ СБОРА ИНФОРМАЦИИ

Специальность: 2.3.5 - «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей»

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель
доктор технических наук, доцент
Антонов Вячеслав Викторович

Уфа - 2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4
1 ТЕОРЕТИЧЕСКИЕ ОСНОВАНИЯ И ЭВОЛЮЦИЯ МЕТОДОВ СЕМАНТИЧЕСКОГО АНАЛИЗА ДАННЫХ 15
1.1 Семантический анализ как междисциплинарная проблема: понятия и задачи 15
1.2 Эволюция парадигм семантического анализа. Актуальные проблемы и перспективные направления 17
1.3 Выводы по главе 1 22
2. МЕТОДОЛОГИЧЕСКИЕ ОСНОВАНИЯ И АРХИТЕКТУРА ГИБРИДНОГО МЕТОДА КВАЛИФИКАЦИИ 24
2.1 Формальная модель процесса квалификации 24
2.2 Сравнительный анализ существующих архитектур гибридных систем квалификации текстов 28
2.3 Структура гибридного метода квалификации 32
2.4 Программно-алгоритмическая реализация и оценка эффективности 39
2.5 Выводы по главе 2 42
3. РАЗРАБОТКА МЕТОДА СЕМАНТИЧЕСКОГО АНАЛИЗА ДАННЫХ 44
3.1 Семантический анализ: понятие, функции, применение в средствах автоматизированной обработки данных 44
3.2 Разработка алгоритма для обработки данных: 49
3.2.1 Базовый алгоритм BPE применяемый для обработки текстовых данных .. 49
3.2.2 Базовые алгоритмы семантического и тонального анализа данных 53
3.3 Алгоритм квалификации неструктурированных данных 57
3.3.2 Базовое представление алгоритма 57
3.3.3 Математическое представление алгоритма 60
3.4 Выводы по главе 3 79
4 СТРУКТУРА ПРОГРАММНОГО РЕШЕНИЯ КВАЛИФИКАЦИИ И ЕЕ РЕАЛИЗАЦИЯ В ПАРСИНГОВОЙ СИСТЕМЕ 80
4.1 Анализ объекта внедрения и существующий механизм квалификации контента 80

4.2 Реализация программного решения и сценариев развертывания 88
4.2.1 Реализация подключаемого программного модуля 88
4.2.2 Моделирование процесса внедрения библиотеки 95
4.3 Механизм взаимодействия с платформами сбора данных 100
4.4. Выводы по главе 4 105
5 ЭКСПЕРИМЕНТАЛЬНАЯ ОЦЕНКА ЭФФЕКТИВНОСТИ МЕТОДА КВАЛИФИКАЦИИ 107
5.1 Методика оценки и характеристика тестовых данных 107
5.2 Экспериментальная оценка качества квалификации контента 111
5.3 Выводы по главе 5 121
ЗАКЛЮЧЕНИЕ 122
СПИСОК СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 126
СПИСОК ЛИТЕРАТУРЫ 128
ПРИЛОЖЕНИЕ А 144
ПРИЛОЖЕНИЕ Б 149
ПРИЛОЖЕНИЕ В 152
ПРИЛОЖЕНИЕ Г 154
ПРИЛОЖЕНИЕ Д 185
ПРИЛОЖЕНИЕ Е 200

ВВЕДЕНИЕ

Современная научная практика характеризуется противоречивой ситуацией, в которой рост вычислительных мощностей и объема научных публикаций сопровождается распадом двух фундаментальных принципов организации науки - воспроизводимости практических результатов и накоплением знания. Это явление обозначается в литературе как «кризис репликации». Его суть заключается в том, что значительная доля исследований, преимущественно в когнитивных, социальных и биомедицинских дисциплинах, не подтверждается в ходе независимой экспертизы.

Параллельно с кризисом репликации обостряется проблема фрагментации данных, проявляющаяся в распределении научных результатов по множеству каналов: часть из них размещаются в коммерческих журналах с платным доступом, часть - в открытых препринтных архивах, сырые датасеты размещаются на выделенных серверах, а исполняемый код - в репозиториях систем контроля версий. Отдельную сложность представляет систематическое исключение из научного оборота нулевых и отрицательных результатов. Подобное рассеивание информации порождает не только операциональные неудобства, но и системную непрозрачность исследовательского процесса, проявляющуюся в замедлении верификационных процедур, блокировке возможностей проведения анализа и синтеза знаний, а также в снижении общей эффективности научной деятельности вследствие дублирования усилий.

Для решения этих проблем необходимо перестроить систему взаимодействия с данными. На сегодняшний день главное это документ, содержаший информацию, однако для машинной обработки необходима понятная цифровая среда. В этом случае также небходима надежная система идентифкации всех субъектов процесса, начиная от авторов документа, заканчивая теми кто с ним взаимодействует, а также классификация того что содержится внутри документа.

Существуют решения которые частично закрывают эти вопросы. Например, ORCID решает проблему, когда у разных исследователей одинаковые имена или

когда они меняют место работы. Идентификатор DOI гарантирует, что ссылки на работы всегда будут работать и не исчезнут. стема Software Heritage (SWHID) сохраняет версии программного кода, чтобы любой мог повторить вычислительный эксперимент. Реестры ROR и идентификаторы грантов помогают увидеть, кто финансировал исследование и какие результаты из этого вышли. Их вазимосвязь складывается в понятный граф: статья ссылается на набор данных, автор подтверждается через ORCID, организация к которой он принадлежит через ROR.

Современный рост неструктурированных данных требует перехода от традиционной публикации к семантически аннотированным объектам. Это позволяет рассматривать исследование не как статичный документ, а как динамический интерфейс.

При таком подходе публикация интерпретируется как интерфейс, объединяющий размеченный текст, визуализации и исполняемые модули. Ключевые сущности снабжаются ссылками на онтологии. Это реализует концептуальный поиск: система находит соответствия между исследованиями даже при расхождениях в терминологии. Метаданные теперь аккумулируют сведения об условиях экспериментов. Сами данные признаются самостоятельным научным продуктом, а не иллюстрацией к тексту. Метаописания при этом должны отвечать принципам FAIR (поиск, доступ, интероперабельность, повторное использование).

Интеграция простых идентификаторов и семантической структуры могла бы позитивно воздействовать на кризис репликации. Связанные идентификаторы образуют граф знаний, который позволяет проследить развитие научной гипотезы. Семантическая разметка смещает поиск с ключевых слов на сущности и отношения между ними. Однако внедрение такой системы - это комплексная социотехническая проблема. Основные проблемы внедрения вэтом случае лежат не только в технологиях, но в устройстве каждой отдельной взятой научной организации.

В прикладном аспекте структурирование начинается с категоризации. Массив разделяют по признакам: тематика, тип источника, статус обработки. Каждой категории присваивают имя, отражающее суть информации. Затем формализуют отношения: вложение, соподчинение или множественная принадлежность. Выбор между жесткой иерархией и сетевой моделью зависит от массива. Древовидные структуры подходят для стабильной информации, гибридные и теговые модели - для динамичных областей.

При работе с мультимодальными данными требует расширения принципов идентификации, квалификации и структурирования за пределами текста. Ключевая сложность заключается в создании цифрового образа для объектов без простой разбивки на части: изображений, видео, симуляций. Решение лежит в сочетании двух инструментов: с помощью векторных представлений создаются устойчивые описания, а онтологии объединяют их в логические пространства.

В рамках исследования под процессом объединения в логические пространства понимается процесс «квалифкации» данных. Данный термин используется в более широком смысле. Он охватывает не только категоризацию, но и концептуальное упорядочивание неструктурированных данных, получаемых из интернет-сообществ. При этом для такого типа данных важно учитывать пользовательский стиль публикаций, а также ее тональность. Для этого могут быть использованы методы стилометрии, обычно включающее статистический анализ и относящееся к письменному тексту. Основой любого стилометрического анализа служит извлечение из текста измеримых характеристик - стилометрических признаков или маркеров. Анализ тональности также необходим для квалификации публикаций гле необходимо распознавание агрессивного поведения, оскорблений и неприемлемого контента. Часто он используется для разработки виртуальных помощников: так можно обучить ассистентов анализировать тональность текста и подстроиться под эмоциональную окраску собеседника.

Однако стилометрический анализ сталкивается с проблемой зависимости от объема текста, классические метрики устойчиво работают на текстах от 1000 слов, тогда как на коротких сообщениях частотные оценки становятся статистически

ненадёжными. Тональный анализ, в свою очередь, ограничен контекстуальной зависимостью эмоциональной окраски: словари задают статические оценки, тогда как значение слова меняется в зависимости от окружения, отрицания, иронии и сарказма. При этом сарказм и ирония остаются наиболее трудными для автоматического распознавания, поскольку требуют выхода за пределы буквального значения.

Инструментарий позволящий частично закрыть выявленные проблемы может быть построен на сочетании нейросетевого восприятия и онтологической логики. Он позволит активно обрабатывать данные, находить скрытые закономерности и строить проверяемые модели. Дополнительные сложности создают пользователи, которые публикуют текстовый контент. Один и тот же пользователь может менять стиль в зависимости от того в какому интернет - сообществе он публикуется, также высокий порог подражания другим

Теоретический фундамент современных методов классификации текстов составляет комплекс работ по формальным языкам и автоматам [1-3], представлению знаний [4], дистрибутивному анализу [5] и основам машинного обучения [6], включая наивный байесовский классификатор [7], метод опорных векторов [8] и статистическую теорию обучения [9]. Развитие нейровероятностных языковых моделей [10] и методов построения векторных представлений слов [11-13] обеспечило переход к нейросетевым подходам, которые дополнительно обогащены исследованиями в области стилометрии [14-17], субсловных единиц [18, 19], обзорами методов атрибуции авторства [20, 21], а также анализом социальных медиа [22, 23], стилистической классификацией [24, 25] и архитектурами глубокого обучения, включая сверточные [26], рекуррентные [27] сети и трансформеры [28], предобученные модели [29, 30], многозадачное обучение [31] и диалоговые системы [32].

Прикладные задачи патентной классификации стимулировали разработку специализированных текстовых представлений [33], гибридных нейросетевых методов [34], моделей на основе дистрибутивных семантик [35] и обзорных исследований [36-39], а также мультимодальных [40, 41] и графовых [42, 43]

подходов; для классификации текстов предложены фреймворки без размеченных данных [44] и обзоры для корпоративных систем [45] и предобученных моделей [46]. Исторические аспекты машинного перевода [47] нашли отражение в современных учебниках [48-51], а нейробиологические модели [52, 53] дополнили арсенал методов обучения с шумными метками [54]. Графовые сверточные сети [55], медицинские приложения LLM [56] и голистические обзоры [57] соседствуют с мультимодальными [58, 59] и унифицированными архитектурами [60], анализом тональности [61], мультимодальными представлениями [62], обогащением векторных моделей субсловной информацией [63] и методами интерпретируемости [64-66], федеративным обучением [67].

Русскоязычные исследования охватывают иерархическое многозадачное обучение [68], символьные сверточные сети [69], обзоры алгоритмов [70], практические руководства по CNN [71] и комплексные обзоры глубокого обучения [72], а также композициональные представления [73] и монографии по интеллектуальному анализу [74-77]; задачи классификации по жанрам [78], категоризации событий и факторов [79], анализа тональности [80], сравнения методов для новостных текстов [81], генерации тестов [82] и аннотации семинаров [83] дополнены сравнительным анализом векторизации [84]. Фундаментальные модели, такие как обработка языка «почти с нуля» [85], тематическое моделирование [86], краудсорсинговые лексиконы [87], вероятностная интерпретация классификаторов [88], многозадачное обучение [89] и исторические модели роста [90], лежат в основе исследований сарказма [91-95], реферирования [96], извлечения терминов [97], семантического анализа [98], сравнения моделей классификации [99], графовых обзоров [100], материалов конференций [101], классификации научных текстов [102], словарных методов тональности [103], сегментации документов [104], разметки текстов о жизненных трудностях [105], обзоров LLM для классификации [106], библиометрического анализа [107], туристических приложений [108] и классических работ по классификации [109]. Современные обзоры диалоговых агентов [110] и многозадачного обучения [111], иерархическая классификация [112], лингвистические аспекты [113], платформы

мониторинга социальных медиа [114, 115], техническая литература по языкам программирования [116-120], нормативно-правовые акты [121-129], статистические методы [130], вопросы защиты сбора данных и открытые наборы данных [131] составляют эмпирическую и методологическую базу для проведения экспериментальных исследований в области классификации текстов.

Таким образом, задача, решаемая в диссертационной работе, является новой и заключается в разработке моделей, методов и программных средств квалификации семантической информации для анализа и интерпретации динамичных массивов пользовательского контента в социальных медиа, обеспечивающих повышение точности обработки данных в системах сбора информации при сохранении интерпретируемости и адаптивности к изменениям данных.

Объектом исследования являются неструктурированные текстовые данные, извлекаемые из открытых интернет-источников, и процессы их автоматизированной обработки.

Предмет исследования: модели, методы и алгоритмы семантической квалификации контента, основанные на комбинации нейросетевых подходов и онтологического моделирования, направленные на повышение точности обработки данных в системах сбора информации.

Целью диссертационной работы является повышение точности классификации (квалификации) неструктурированных текстовых данных за счет разработки гибридного метода, интегрирующего контекстуальные векторные представления, тональный и стилометрический анализ, а также его программная реализация в виде модуля для систем сбора информации.

Цель исследования достигается через последовательное решение теоретических и прикладных задач.

На основе сравнительного анализа существующих методов были определены следующие задачи:

1. Разработать формальную модель квалификации неструктурированных данных (п. 1 паспорта специальности).

2. Разработать алгоритм квалификации контента интернет-источников на базе предложенной модели (п. 3 паспорта специальности).

3. Спроектировать программное решение системы квалификации и реализовать её ключевые модули (п. 3 паспорта специальности).

4. Провести внедрение предлагаемой модели и алгоритма квалификации в действующую систему автоматизированного сбора данных и экспериментальную оценку их эффективности с формированием рекомендаций по дальнейшему развитию (п. 10 паспорта специальности).

На защиту выносятся следующие положения:

1. Формальная модель квалификации неструктурированных данных, построенная на трехуровневой структуре «признак → категория → сет» и механизме множественной принадлежности семантических единиц. Модель учитывает контекстную зависимость классификации в условиях полисемии (п. 1 паспорта специальности).

2. Алгоритм гибридной квалификации текстового контента, основанный на совместном применении контекстуализированных векторных представлений, тонального анализа и стилометрической метрики (дельта Берроуза). Такой подход позволяет выявлять аномальный контент и повышает точность категоризации (п. 3 паспорта специальности).

3. Программное решение, реализующее предложенный алгоритм. Программная структура поддерживает гибкие сценарии развертывания и включает механизм динамической маршрутизации потоков данных, обеспечивающий семантическую согласованность и масштабируемость (п. 3 паспорта специальности).

4. Результаты внедрения и экспериментальной апробации предлагаемого решения в составе промышленной информационной аналитической системе. Апробация подтверждает повышение точности квалификации контента и практическую значимость разработанных решений для задач мониторинга информационного пространства.

Научная новизна диссертационного исследования состоит в следующем:

1. Разработана формальная модель квалификации с иерархией «признак → категория → сет», реализующая вместо одного уровня принадлежности множественную принадлежность, то есть один объект может одновременно относиться к нескольким категориям. В отличие от классических онтологических моделей с жёсткой привязкой к одной категории, предложенная модель учитывает полисемию и контекстную зависимость смысла коротких зашумлённых сообщений. Модель позволяет формализовать контекстную многозначность и снизить число ошибок категоризации коротких сообщений, где смысл размыт и зависит от контекста.

2. Предложен алгоритм, в котором комплексно объединены контекстуализированные векторные представления, тональный анализ и адаптирована стилометрическая метрика (дельта Берроуза) для коротких текстов социальных медиа. В отличие от существующих подходов, где стилометрические и семантические признаки объединяются на уровне решений, в предложенном алгоритме интеграция выполняется на уровне признакового пространства. Решение позволяет повысить точность квалифкациии неструктрированных данных и обеспечить выявление аномального контента при сохранении интерпретируемости решений.

3. Предложено использование порядковой классификации с обучаемыми порогами, позволяющий оценивать степень серьезности контента независимо от категориального решения. в отличие от традиционных категориальных моделей, которые дают только номинальную метку, предложенный подход раздельно и одновременно упорядоченно оценивает степень значимости контента относительно темы; комбинация категориальной классификации и порядковой регрессии позволяет получать как категорию, так и уровень тяжести, не смешивая эти задачи. Такой подход обеспечивает существенное согласие с экспертами и позволяет приоритизировать обработку контента без переобучения категориальной модели.

В совокупности предложенные решения повышают точность квалификации контента, сохраняя при этом интерпретируемость и адаптивность к изменениям данных. Это составляет существенный вклад в развитие методов обработки естественного языка и построения масштабируемых систем анализа потоковых данных.

Теоретическая значимость работы заключается в развитии теоретических основ автоматической классификации неструктурированных текстовых данных:

1. Предложена формальная модель квалификации, которая, в отличие от существующих подходов с плоскостной классификацией, вводит иерархическую структуру и механизм множественной принадлежности семантических единиц. Это развивает представления об учете контекстной зависимости и полисемии при категоризации коротких сообщений в динамичных потоках данных социальных медиа.

2. Разработан гибридный метод семантического анализа, теоретически обосновывающий комбинирование контекстуализированных векторных представлений (на базе архитектуры трансформер), тонального анализа и стилометрической метрики для повышения точности квалификации. Метод расширяет возможности классических статистических и нейросетевых подходов, обеспечивая выявление аномального контента на основе статистических отклонений частотных характеристик.

3. Предложена программная структура реализации с механизмом динамической адаптации маршрутизации потоков данных в зависимости от семантики и аномальности контента. Это вносит вклад в теоретические основы построения масштабируемых систем обработки потоковых данных.

Практическим результатом исследования является программное решение квалификации неструктурированных данных, реализующий разработанные методы и подтверждающий их эффективность. Результаты работы вносят вклад в развитие технологий обработки естественного языка за счет:

— повышения точности квалификации контента;

— ускорения анализа данных за счет распараллеливания;

— учета сложных семантических связей в тексте;

— улучшения контекстного и тонального анализа;

— создания более эффективных систем для понимания естественного языка.

Методология и методы исследования. В ходе работы над решением поставленных задач использовались модели и методы машинного обучения, теория суперпозиции, исследования Хомского в области семантического анализа, методология системного анализа, онтологическое моделирование, теория графов и сетевой анализ, теория множеств, методы векторных представлений, сравнительный анализ и методы экспертной оценки.

При практической реализации полученных теоретических результатов использовались методы объектно-ориентированного программирования.

Апробация и реализация результатов

Основные положения и результаты диссертационного исследования докладывались и обсуждались на следующих научных конференциях и семинарах:

– XVIII Всероссийская молодёжная научная конференция «Мавлютовские чтения», Уфа, 2024;

– XVII Всероссийская зимняя школа-семинар магистрантов, аспирантов и молодых ученых «Актуальные проблемы науки и техники», Уфа, 2024

– XIX Всероссийская молодёжная научная конференция «Мавлютовские чтения», Уфа, 2025;

– XVIII Всероссийская зимняя школа-семинар магистрантов, аспирантов и молодых ученых «Актуальные проблемы науки и техники, Уфа, 2025];

– научных семинарах кафедры Автоматизированных систем управления Уфимского университета науки и технологий (2024-2026 гг.).

Разработанный метод квалификации неструктурированных данных и программное решение, реализующий предложенные алгоритмы, внедрены и используются в составе промышленной информационно-аналитической системы для выявления деструткивного контента, для модерации кконтента и в учебном процессе, что подтверждено соответствующими актми о внедрении (Приложение А).

Публикации. Результаты исследований получили отражение в публикациях автора в научных периодические изданиях, которые включены в Российский индекс научного цитирования [132-138]. Наиболее значимые результаты, выносимые на защиту, получили отражение в следующих научных публикациях автора, в журналах входящих в перечень утверждённый Высшей аттестационной комиссией [139-142]. Также автором разработаны несколько программ для ЭВМ и схема баз данных, на которые получены свидетельства о государственной регистрации [143 - 145].

Личный вклад автора. Автором диссертационного исследования непосредственно выполнен полный цикл работы: формулировка положения и теоретических выводов, проектирование структуры программного средства, разработка алгоритмов, создание модели, проведение экспериментов и координация опытной эксплуатации модуля квалификации неструктурированных данных с последующей аналитикой его эффективности.

Структура и объём работы: диссертация объемом 200 машинописных страниц, содержит введение, пять глав и заключение, список литературы (147 наименований), 46 формул, 19 таблиц, 24 рисунка, 6 приложений.

В работе разработаны математические модели, алгоритм и программная реализация, относящиеся непосредственно к математическому и программному обеспечению систем обработки информации. В отличие от исследований в области системного анализа (специальность 2.3.1), где акцент делается на принятие решений человеком, и управления в организационных системах (специальность 2.3.4), где рассматриваются социально-экономические и управленческие аспекты, представленная работа сосредоточена на алгоритмических и программных аспектах автоматизированной обработки неструктурированных текстовых данных. Основным результатом исследования является разработка комплекса математического и программного обеспечения для квалификации контента в системах сбора информации, что непосредственно соответствует области исследований, определенной паспортом специальности 2.3.5.

1 ТЕОРЕТИЧЕСКИЕ ОСНОВАНИЯ И ЭВОЛЮЦИЯ МЕТОДОВ СЕМАНТИЧЕСКОГО АНАЛИЗА ДАННЫХ

1.1 Семантический анализ как междисциплинарная проблема: понятия и задачи

Информацию в цифровой среде принято рассматривать через три операции: хранение, передачу и практическое применение. В условиях цифрового общества она приобретает свойства экономического блага - ею обмениваются, её защищают, преобразуют. Но любое преобразование начинается с получения. Современные методы получения информации автоматизированы: это сбор массивов, интеллектуальный поиск с машинным обучением, интеграция систем сбора данных через API.

Качество конечного информационного продукта зависит не только от сбора, но и от того, как организованы данные. Ключевые операции здесь - структурирование, идентификация и квалификация. Они взаимосвязаны и образуют иерархию.

Идентификация информации - это процесс, в ходе которого каждой единице данных присваивают уникальный набор признаков. Цель - создать цифровой образ, позволяющий отличить один элемент от другого и точно адресовать его в системе. Идентификация превращает анонимный фрагмент в узнаваемый объект с чёткими границами. Процедура начинается с записи присущих обекту параметров: источник, авторство, ключевые понятия, тематика формальные атрибуты (тип файла, объём), состояние (версия, статус обработки), позиция в логических последовательностях. Затем набор параметров преобразуется в идентификаторы, понятные машинам. От того насколько преобразование выполнено однозначно зависит работа поиска,фильтрации и аналитической обрабокти.

Для семантического анализа динамических потоков данных ключевой задачей выступает квалификация (категоризация) - отнесение неструктурированного контента к заданным классам. Однако эффективное решение этой задачи требует не только алгоритмов классификации, но и

адекватной модели представления самих связей между объектами. Связи могут быть иерархическими, последовательными (временная или процессуальная логика), либо сетевыми (смысловое сходство). Выбор модели определяет, насколько гибко система сможет отражать сложные взаимосвязи предметной области и обрабатывать новые данные без полной перестройки архитектуры.

Иерархические структуры обладают принципиальными ограничениями: они не позволяют описывать перекрестные связи между элементами из разных ветвей, а изменение вышестоящего узла требует перестройки всех подчинённых структур. Эти ограничения снимаются в сетевых моделях (графах), где любой узел может быть связан с произвольным числом других узлов, а корневой элемент не обязателен. Переход к таким моделям означает отказ от жесткого упорядочивания в пользу гибких форм, способных отражать реальную многомерность знаний.

Эволюция вычислительных подходов к работе с текстом показывает, почему простые системы оказались недостаточны. Первые системы машинного перевода 1950-х годов (Джорджтаунский эксперимент, группа И.А. Бельской на БЭСМ) строились на двуязычных словарях и жестких грамматических правилах. Они демонстрировали работоспособность лишь на на небольших наборах текстовых данных и не справлялись с контекстной многозначностью. Диалоговая система ELIZA (1966) обходила эту проблему: она не пыталась понять смысл, а сопоставляла фразы с шаблонами. Это позволяло имитировать диалог, но не решало задачу семантического анализа как такового.

Таким образом, к концу XX века стало очевидно: ни словарные правила, ни шаблонные реакции не обеспечивают гибкой квалификации текстов. Требовались модели, способные удерживать контекст и выявлять скрытые семантические связи. На современном этапе эту функцию берут на себя нейросетевые архитектуры, а теоретическим базисом для формализации связей служат онтологии и семантические сети. Именно их сочетание позволяет сегодня решать задачу квалификации не как подбор однозначного значения, а как размещение контента в смысловом пространстве с множественными пересекающимися отношениями.

1.2 Эволюция парадигм семантического анализа. Актуальные проблемы и перспективные направления

Развитие методов семантического анализа и классификации текстов проходит через несколько этапов. Каждый следующий этап преодолевает ограничения предыдущего, но одновременно порождает новые задачи, требующие решения. Рассмотрение этих этапов необходимо для обоснования цели данного исследования - разработки гибридного метода квалификации потоков неструктурированного пользовательского контента.

Анализ патентов в области обработки естественного языка показывает, что за последние десять лет число патентных заявок стабильно растёт [6]. Основными заявителями выступают крупные технологические корпорации и исследовательские центры. Эволюция патентных решений повторяет общую траекторию развития методов: от статистических (TF-IDF) и классификационных (SVM) подходов к нейросетевым и гибридным моделям [146].

Первым этапом может считать символьный: формализация языка. Семантический анализ как научное направление сформировался на стыке лингвистики, математической логики и теории вычислений. В работе «Синтаксические структуры» [1] Н. Хомский предложил трансформационно-порождающую грамматику и иерархию формальных грамматик (типы 0-3), заложив математические основы для алгоритмической обработки языка. С. Клини разработал теорию регулярных выражений [2], а работы Дж. Хопкрофта по теории автоматов [3] создали базу для построения синтаксических анализаторов, М. Мински предложил фреймовую модель представления знаний, основанную на иерархической структуре слотов и условий соответствия [4].

Символьные подходы точны в узких, формализованных областях. За пределами этих областей проявляются два ограничения. Первое - трудоёмкость построения баз знаний: создание онтологии требует длительной экспертной работы, любое расширение выполняется вручную, масштабирование практически невозможно. Второе - неспособность обрабатывать вариативность естественного языка. Сленг, сокращения, намеренные искажения, эмодзи, ирония и сарказм -

явления, не описываемые конечным набором правил. Полисемия и омонимия также остаются неразрешимыми: выбор значения в коротком сообщении требует либо огромного числа контекстных правил, либо знаний о мире, что выходит за рамки символьного подхода.

Следующий этап - статистический. Фокус сместился с содержательного анализа на выявление закономерностей в больших массивах данных. Концептуальную основу составил принцип дистрибутивной лингвистики З. Харриса [5], согласно которому значение слова определяется его окружением. Т. Митчелл [6] дал классическое определение обучения как улучшения качества выполнения задачи с приобретением опыта.

Первые отечественные системы категоризации («Дискант», «Вега») опирались на частотные словари и простые правила, не учитывали контекст. Наивный байесовский классификатор [7], исходящий из условной независимости признаков, показывает приемлемую точность на многих задачах, но теряет качество там, где зависимости между словами становятся определяющими. Метод опорных векторов (SVM), предложенный В. Вапником [9] и адаптированный для текстов Т. Йоахимсом [8], обеспечил устойчивость к большому количеству признаков и способность к обобщению.

Ограничение статистических методов лежит в модели «мешка слов»: текст превращается в неупорядоченное множество частот. Порядок слов, синтаксические связи, фразеологизмы и идиомы игнорируются. Предложения с разным порядком слов не различаются, омонимия не распознаётся. Для каждой предметной области требуется ручное конструирование признаков, что препятствует переносу моделей.

Переход от дискретных признаков к непрерывным векторам стал возможен с развитием нейросетей и ростом вычислительных мощностей. Й. Бенжио [10] предложил нейронную вероятностную языковую модель, где каждому слову сопоставлялся вектор, а вероятность последовательности оценивалась сетью, обученной предсказывать следующее слово. Модели Word2Vec [11], реализованные в архитектурах CBOW и Skip-gram, позволили получать векторные представления слов на больших корпусах за приемлемое время. В отличие от них,

GloVe [12] опирается на глобальную статистику совместной встречаемости слов. В работе Леви и Гольдберга [13] показано, что Skip-gram с отрицательной выборкой может быть интерпретирован как разложение матрицы, показывающей, насколько слова связаны друг с другом. Авторы GloVe [12] указывали на то что статические эмбеддинги закрепляют за каждым словом один вектор. У любого слова будет ровно один вектор, усреднённый по всем контекстам в обучающем корпусе. Кроме того, эмбеддинги остаются неизменными во времени - они отражают состояние языка на момент формирования обучающей выборки и не учитывают появление новых слов и значений без полного переобучения модели. Для анализа потоков данных из сети Интернет это ограничение является существенным.

С развитием глубокого обучения появились архитектуры, учитывающие структуру текста. Й. Ким [26] предложил использовать одномерные свёртки с фильтрами разного размера для тогочтобы выделить словосочетания из нескольких слов. Позже Р. Джонсон и Т. Чжан [27] использовали LSTM (долгая краткосрочная память) для учёта локального контекста. Также была предложена гибридная модель BiLSTM-GCN, сочетающая двунаправленную память с графовыми свёрточными сетями; на наборе THUCNews достигнута F1-мера 91,37%.

CNN улавливают только локальные зависимости в пределах рецептивного поля. LSTM способны удерживать дальний контекст, но последовательные вычисления ограничивают масштабируемость. Эти противоречия стимулировали поиск архитектур без рекуррентных и свёрточных ограничений.

В 2017 году группа исследователей под руководством А. Вашвани [28] предложила архитектуру трансформер, отказавшись от рекуррентных и свёрточных слоёв. Предложенный механизм внимания вычисляет взвешенную сумму векторов-значений; вес определяется совместимостью ключа с запросом. Каждая позиция взаимодействует с любой другой независимо от расстояния, вычисления параллелизуются.

На базе трансформера сложились две ветви. Первая это BERT [29], она использует кодирующую часть модели и учится на двух задачах: восстанавливает пропущенные слова и предсказывает, какое предложение идёт следующим. Вектор

слова (токена) меняется в зависимости от окружения - и это решает проблему многозначности. Второе - большие языковые модели (GPT и похожие). Они построены на декодирующей архитектуре и учатся предсказывать следующий токен.

Патентные исследования отражают доминирование трансформерной парадигмы. Современные патенты [36] посвящены применению BERT для задач классификации, гибридным архитектурам и мультимодальному анализу [30, 31, 32]. Энкодерные модели эффективнее больших языковых моделей (LLM) по времени вывода и энергопотреблению. При этом LLM показывают лучший результат на редких классах [33]. Для различных языковых групп создаются лингвоспецифичные модели на основе трансформерной архитектуры [34, 36].

Архитектура трансформеров позволила решить проблему контекстуального понимания текста, однако породила новые сложности. К ним относятся: высокая ресурсоемкость обучения и вывода, статичность обучения. Кроме того, генеративные модели склонны к фактическим ошибкам, уязвимы к смещениям, то есть перенимают предвзятость и перекосы, которые есть в данных.

Перечисленные ограничения особенно значимы для задач квалификации динамических потоков пользовательского контента из социальных медиа, новостных лент и диалоговых платформ. Такой контент характеризуется лингвистическим шумом: орфографическими ошибками, намеренными искажениями, неологизмами, сленгом, эмодзи и «мемами», несущими смысловую нагрузку. Короткие сообщения обладают высокой многозначностью при минимальном вербальном окружении. Язык в интернет-среде эволюционирует быстро: постоянно появляются новые термины, заимствования, меняются тематические тренды. Один и тот же текст может получать разные интерпретации в зависимости от сообщества, региона или платформы.

Ни один из рассмотренных подходов в отдельности не обеспечивает одновременного выполнения требований классификаций неструктированныйх данных Интернет-источников: обработка в реальном времени, интерпретируемость решений, адаптивность к изменениям данных и устойчивость к шуму. Жесткая

однозначная классификация недостаточно эффективна для социальных медиа, где смысл отдельно взятой публикации определяется контекстом, а одно высказывание может относиться к нескольким темам. Более предпочтительными являются иерархические и многомерные модели с частичной принадлежностью к категориям. Необходима гибридизация методов, объединяющая семантическую глубину трансформеров с интерпретируемостью онтологических структур и вычислительной эффективностью статистических подходов.

Из чего можно следует что цель исследования - повышение точности классификации (квалификации) неструктурированных текстовых данных за счет разработки гибридного метода, интегрирующего контекстуальные векторные представления, тональный и стилометрический анализ, а также его программная реализация в виде модуля для систем сбора информации.

Научная гипотеза заключается в том, что гибридизация символических методов представления знаний (онтологий) с субсимволическими моделями глубокого обучения на основе трансформеров позволяет преодолеть ограничения чистых нейросетевых моделей - низкую интерпретируемость, непрозрачность принятия решений и недостаточную чувствительность к предметно-специфическим ограничениям. Интеграция онтологических структур в архитектуру классификатора обеспечивает семантическую маршрутизацию запросов и снижает вычислительную сложность за счет распределения нагрузки в зависимости от лингвистической сложности входного текста.

Для того чтобы подтвердить гипотезу необходимо решить следующие задачи:

1. Разработать формальную модель квалификации неструктурированных данных, учитывающей контекстную многозначность, специфику интернет-коммуникаций (лингвистический шум, полисемия в коротких сообщениях, концептуальный дрейф) и допускающей иерархическое представление «признак → категория → сет».

2. Разработать алгоритм гибридной квалификации контента интернет-источников на основе комбинации контекстуализированных векторных представлений, методов тонального анализа и стилометрических метрик.

3. Спроектировать прогроммное решения системы квалификации и реализации ключевых модулей. Оно должно обеспечить динамическую маршрутизацию потоков данных для масштабируемой обработки в режиме, близком к реальному времени. Для этого необходимо предусмотреть разработку библиотеки, которая предоставляет стандартизированный интерфейс для интеграции с существующими системами сбора данных.

4. Внедрение полученного решения в действующую инфраструктуру и его опытная эксплуатация. На этом этапе проводится оценка эффективности - по точности классификации, производительности, интерпретируемости результатов и адаптивности к изменению входных потоков. На основе полученных данных формулируются рекомендации по доработке метода и его дальнейшему практическому применению.

Проведённый анализ эволюции методов семантического анализа и патентной активности в области обработки естественного языка показал устойчивый переход от статистических и классификационных подходов к нейросетевым и гибридным моделям. Однако ни один из существующих подходов не покрывает весь спектр требований к квалификации коротких зашумлённых сообщений. Это обосновывает необходимость разработки гибридного метода, что и является целью настоящей работы.

1.3 Выводы по главе 1

Проведённый в первой главе анализ семантического анализа как междисциплинарной проблемы и эволюции его парадигм позволил выявить ключевые понятия, задачи и ограничения существующих подходов, а также определить перспективные направления развития методов квалификации неструктурированных текстовых данных.

1. Семантический анализ как научное направление сформировался на стыке лингвистики и математики. Ключевыми операциями обработки информации являются идентификация, структурирование и квалификация. Однако словарные правила и шаблонные реакции не обеспечивают гибкой квалификации текста. Из чего следует вывод, что требуются модели, удерживающие контекст и выявляющие скрытые семантические связи. На современном этапе эту функцию выполняют нейросетевые архитектуры.

2. Анализ эволюции методов семантического анализа и патентной активности в области обработки естественного языка показал устойчивый переход от статистических и классификационных подходов к нейросетевым и гибридным моделям. Выявлены ограничения каждого этапа: символьные подходы точны и интерпретируемы, но не масштабируемы; статистические - масштабируемы, но теряют контекст и порядок слов; статические эмбеддинги частично восстанавливают семантику, но не работают с многозначностью; рекуррентные сети учитывают порядок, но медленны; свёрточные - быстры, но захватывают лишь локальный контекст; трансформеры обеспечивают глубокое контекстуальное понимание, но ресурсоёмки и статичны при концептуальном дрейфе.

Для задачи квалификации потоковых данных из интернет-источников ни один из существующих подходов в отдельности не является достаточным: пользовательский контент социальных медиа характеризуется высокой плотностью лингвистического шума (орфографические ошибки, сленг, эмодзи, мемы), высокой полисемией коротких сообщений, быстрой эволюцией языка, культурной и субкультурной спецификой.

На основе выявленных ограничений и перспективных направлений в диссертационной работе поставлены задачи по разработке формальной модели квалификации с иерархией «признак → категория → сет» и механизмом множественной принадлежности семантических единиц, созданию гибридного алгоритма на основе трансформерных архитектур, тонального анализа и стилометрических метрик, проектированию программной структуры для обработки данных в режиме, близком к реальному времени, а также внедрению

решения в действующую систему автоматизированного сбора данных с оценкой эффективности по точности, производительности, интерпретируемости и адаптивности к изменениям данных.

2. МЕТОДОЛОГИЧЕСКИЕ ОСНОВАНИЯ И АРХИТЕКТУРА ГИБРИДНОГО МЕТОДА КВАЛИФИКАЦИИ

2.1 Формальная модель процесса квалификации

Онтология задает правила для категорий и признаков, обеспечивая интерпретируемость результатов квалификации. В данном исследовании под онтологией понимается формальная спецификация предметной области: выстраивается разбиение на классы, идентифицируются релевантные классы, описываются свойства и фиксируются связи между ними. Для логической непротиворечивости вводятся ограничения - они определяют, какие комбинации категорий допустимы, а какие нет.

Онтологический подход ценен для квалификации текстов социальных медиа: он дает единый словарь с иерархией «общее - частное», позволяет ранжировать понятия по важности и задавать точные критерии значимости контента. Однако, как отмечалось в первой главе, чистые онтологии негибки и плохо адаптируются к эмпирическому разнообразию. Поэтому предлагается использовать онтологию не как изолированный классификатор, а как компонент гибридной архитектуры. Она отвечает за структурную валидацию и интерпретируемость на верхнем уровне системы.

Далее необходимо определить исходные элементы семантического представления, которые используются в анализе. Семантические примитивы - это базовые составляющие языковых высказываний, выражающие элементарные смыслы. Концепция семантических примитивов разработана А. Вежбицкой с целью построения общей семантической теории естественного языка. Задача семантики, согласно замыслу Вежбицкой, «состоит в том, чтобы выявить структуру мысли, скрытую за внешней формой языка». Один примитив нельзя свести к другому. На их основе строятся семантические функции, которые

описывают структурную организацию текста. Каждому лексическому элементу соответствует функция, интегрирующую как базовые примитивы, так и смысл данного элемента в конкретном контексте.

Эти элементы образуют концептуальный базис для трёхуровневой архитектуры гибридного метода (её детальное описание приведено в разделе 2.2). Одна из специфических проблем, которую необходимо решить на уровне формальной модели, - пересечение семантических пространств на уровне синонимичных единиц. Под семантическими пространствами понимается совокупность маркеров с их связями и типологией. Сочетания синонимов могут относиться к одному маркеру, тогда как их единичное употребление - к другому.

Для работы с такими пересечениями вводится понятие «многопринадлежности узла». Оно характеризует взаимосвязанное состояние двух или более элементов. Семантическая единица может одновременно принадлежать к различным категориям. Это описывается через аналогию с суперпозицией состояний: до момента квалификации единица находится в неопределённом состоянии, характеризующемся распределением вероятностей принадлежности к различным категориям; после отнесения к категории семантическое пространство принимает состояние определения.

Формально многопринадлежность семантической единицы может быть представлена следующим образом: пусть $C = \{c_1, c_2, ...c_k\}$ - множество категорий, w - семантическая единица. Тогда состояние единицы до квалификации описывается вектором вероятностей: $P(w) = [p_1, p_2, ...p_k], \sum_{i=1}^{k} p_x = 1, p_x \geq 0$, где p_x - вероятность принадлежности единицы w категории c_x. В процессе квалификации на основе контекстного анализа и извлечённых признаков происходит уточнение данного распределения, и единица может быть отнесена к одной или нескольким категориям с соответствующими весовыми коэффициентами.

Данный подход позволяет более точно моделировать феномены полисемии и контекстной зависимости смысла по сравнению с традиционными методами, которые жёстко закрепляют единицу за одной категорией. В рамках предлагаемой

трёхуровневой архитектуры механизм многопринадлежности реализуется на уровне интеграции и классификации, где свёрточная нейронная сеть принимает решение о категориальной принадлежности с учётом распределения вероятностей, полученного на предыдущих уровнях анализа.

Обобщая изложенное, формальная модель квалификации может быть представлена в виде кортежа: $M = \langle W, C, R, \Omega, \Phi \rangle$,

где:

– $W = \{w_1, w_2, ..., w_n\}$ - множество семантических единиц;

– $C = \{c_1, c_2, ..., c_k\}$ - множество категорий, к которым могут быть отнесены семантические единицы;

– $R = \{r_1, r_2, ..., r_m\}$ - множество отношений между категориями и единицами (иерархические, ассоциативные, логические);

– Ω - онтологическая спецификация, задающая ограничения на допустимые комбинации категорий и признаков;

– $\Phi : W \to \Delta(C)$ - функция квалификации, отображающая множество семантических единиц в пространство категорий, где $\Delta(C)$- множество распределений вероятностей на множестве категорий C, отражающее принцип многопринадлежности. Функция может быть реализована в двух режимах: одноместочный, используется, когда онтология требует единственной категории и многоместочный - каждая категория получает вероятность независимо.

Формирование сета выполянется согласно следующими правилу:$Set = f\ (Cat, \Omega)$,где Cat - множество категорий, Ω - онтологическая спецификация. Сет определяется как максимальный класс онтологии, покрывающий все предсказанные категории и не противоречащий аксиомам. Если такого класса нет, формируется сет «Прочее» с пометкой «Требует подтверждения эксперта».

Функция квалификации Φ реализуется в рамках трёхуровневой архитектуры:

1. На первом уровне (признаковый анализ) формируется множество признаков $F = \{f_1, f_2, \ldots f_l\}$, характеризующих семантические единицы;

2. На втором уровне (семантический анализ) на основе признаков F строится семантическая сеть $G = \langle Q, R \rangle$, где Q - множество вершин, R - множество рёбер (отношений);

3. На третьем уровне (интеграция и классификация) на основе семантической сети G и признаков F вычисляется распределение вероятностей $\Phi(e) \in \Delta(C)$.

Представленная формальная модель создаёт теоретическую основу для разработки алгоритмического и программного обеспечения, описываемого в последующих разделах. На базе данной модели в разделе 2.2 строится трёхуровневая архитектура гибридного метода квалификации, а в разделе 2.3 предлагаются конкретные программно-алгоритмические решения, реализующие функцию квалификации Φ с учётом требований к производительности и масштабируемости при обработке потоковых данных из социальных медиа.

Онтологическая спецификация Ω формализуется как $\Omega = \langle U, P, A, L, \Theta \rangle$ (Таблица 1):

Таблица 1 - Компоненты онтологии

Компонент	**Обозначение**	**Описание**
Классы	$U=\{u_1, u_2, \ldots, u_k\}$	Множество классов онтологии (категории, сеты, степени тяжести)
Свойства	$P=\{p_1, p_2, \ldots, p_l\}$	Описание характеристики классов
Аксиомы	$A=\{a_1, a_2, \ldots, a_t\}$	Множество аксиом, задающих логические ограничения
Лексикон	$L=\{(s_z, c_x)\}$	Множество пар «лексическая единица - класс». S_z - слово, лемма или фраза; c_x - класс онтологии
Правила	$\Theta=\{\theta_1, \theta_2, \ldots, \theta_r\}$	Множество логических правил для проверки и корректировки предсказаний

Валидация результатов нейросети выполняется путем проверки предсказанных атрибутов на соответствие аксиомам и правилам. Например, для текста «Принтер не печатает, срочно!» нейросеть может предсказать следующие значения: сет = Техническая_поддержка, категория = Проблема_с_оборудованием, степень = 4, сарказм = False. Валидатор последовательно проверяет принадлежность сета онтологическому классу (аксиома иерархии), диапазон степени (аксиома ограничения), выполнение правил T1 (наличие ключевых слов «принтер» и «не печатает»), T2 (наличие слова «срочно») и T3 (отсутствие сарказма при высокой степени). Если все проверки пройдены, результат признается валидным. При обнаружении несоответствия - например, когда нейросеть относит текст с упоминанием принтера к сету Запрос_на_настройку или присваивает высокую степень при наличии сарказма - результат либо корректируется, либо направляется на ручную проверку. Это повышает надежность и интерпретируемость модели.

Таким образом модель реализует механизм многопринадлежности, трёхуровневую структуру «признак → категория → сет», что создаёт основу для алгоритмической и программной реализации в последующих разделах.

2.2 Сравнительный анализ существующих архитектур гибридных систем квалификации текстов

Для обоснования выбора архитектуры, предлагаемой в разделе 2.3, проведен сравнительный анализ существующих подходов к построению гибридных систем квалификации (классификации) текстов. Необходимость такого анализа связана с тем, что задача квалификации коротких зашумленных сообщений в социальных медиа предъявляет требования, которые не могут быть полностью удовлетворены ни одним из текущих подходов. Нейросетевые методы обеспечивают глубину понимания, но обладают высокой ресурсоемкостью; статистические методы интерпретируемы и эффективны, но теряют семантику; лингвистические методы логически строги, но неадаптивны к вариативности языка. Анализ выполнен по следующим критериям: точность классификации, интерпретируемость решений,

устойчивость к лингвистическому шуму и полисемии, вычислительная эффективность, адаптивность к изменениям данных и возможность интеграции в потоковые системы обработки.

В предметной области автоматической классификации текстов выделяются три основных типа гибридных архитектур.

Последовательные (конвейерные) системы, в которых выход одного метода служит входом для другого . Такие архитектуры просты в реализации и широко применяются в системах с поэтапным уточнением классификационного решения. Улучшения точности можно добиться путем добавления двух сверточных слоев после рекуррентной обработки [7]. Однако ключевое ограничение конвейерных систем состоит в том, что ошибки, возникающие на ранних этапах, накапливаются и не могут быть исправлены на последующих уровнях, что критично для зашумленных данных социальных медиа.

Современные исследования подтверждают, что гибридные архитектуры, объединяющие трансформеры с рекуррентными или сверточными слоями, демонстрируют высокую эффективность, однако их применение ограничено ресурсоемкостью и сложностью настройки. Гибридные архитектуры, сочетающие механизмы самовнимания с структурированными моделями пространства состояний, могут достичь убедительного баланса между качеством моделирования и вычислительной эффективностью, особенно для задач с длинным контекстом. При этом авторы отмечают, что в литературе отсутствует систематический анализ стратегий гибридизации, что затрудняет понимание относительных преимуществ различных архитектурных решений [147].

Третий тип - иерархические архитектуры с глубокой интеграцией разнородных методов. В работе Мешковой предложен подход к построению гибридного классификатора, совмещающего свойства семантических и ассоциативных нейросетевых парадигм для решения задачи автоматической классификации текста [146]. Предложенная модель состоит из трех иерархически выстроенных слоев, обобщающих первоначальные единицы текста (слова) в понятия, а понятия - в области знаний. Первый слой содержит нейроны-слова,

связанные между собой на основе словарных определений; второй слой - нейроны-понятия, связанные с нейронами-словами, входящими в их определения; третий слой - нейроны-области знаний, являющиеся выходными. Связи между нейронами описываются матрицами смежности, что позволяет представить сеть в виде графа. Автор отмечает, что гибридизация позволяет компенсировать недостатки каждой парадигмы: семантические сети эффективны для выделения смысла текста, но трудоемки в построении; ассоциативные нейронные сети хорошо классифицируют по признакам, но не учитывают смысловые связи. В работе Мешковой также подчеркивается, что разработка гибридных моделей, совмещающих принципиально различные подходы, стала одним из наиболее актуальных направлений, поскольку отсутствие новой принципиальной идеи стимулирует поиск эффективных комбинаций существующих методов.

Таблица 2 - Сравнение архитектур

Критерий	**Последовательные (конвейерные)**	**Ансамблевые (в т.ч. на LLM)**	**Иерархические с глубокой интеграцией**
Точность классификации	Средняя; повышается добавлением сверточных слоёв после рекуррентной обработки	Высокая; трансформеры превосходят рекуррентные модели по точности на длинных текстах	Высокая за счёт глубокой интеграции разнородных методов
Интерпретируемость решений	Не обеспечивают требуемой интерпретируемости для задач, где решение надлежит обосновать	Низкая; внутренняя логика больших языковых моделей непрозрачна	Повышенная: решения прослеживаются через слои «слова → понятия → области знаний»
Устойчивость к лингвистическому шуму и полисемии	Низкая: страдают от накопления ошибок, которые не исправляются на последующих уровнях	Средняя/высокая на длинных текстах, но нестабильна при ограниченных ресурсах	Средняя; зависит от статичности семантических связей

Вычислительная эффективность	Высокая: конвейер прост и дёшев в реализации	Высокая ресурсоёмкость, делающая применение проблематичным в условиях ограниченных ресурсов	Средняя/низкая: сложны в разработке и настройке
Адаптивность к изменениям данных	Средняя	Высокая	Ограничена статичностью семантических связей при концептуальном дрейфе
Интеграция в потоковые системы обработки	Возможна, но с накоплением ошибок	Неприменимы для потоковой обработки в реальном времени из-за ресурсоёмкости	Сложна из-за трудоёмкости разработки и настройки
Ключевое ограничение для задачи квалификации коротких зашумлённых сообщений	Не обеспечивают интерпретируемости и накапливают ошибки	Высокая ресурсоёмкость → неприменимы для потоковой обработки в реальном времени	Сложность разработки и настройки; ограниченная адаптивность к концептуальному дрейфу

Анализ показывает, что каждый из рассмотренных типов архитектур имеет свои ограничения применительно к задаче квалификации коротких зашумленных сообщений в социальных медиа. Конвейерные системы, как правило, не обеспечивают требуемой интерпретируемости для задач, где решение надлежит обосновать, и страдают от накопления ошибок. Ансамблевые системы, особенно на основе больших языковых моделей, обладают высокой ресурсоемкостью, что делает их неприменимыми для потоковой обработки в реальном времени; при этом исследования показывают, что трансформеры превосходят рекуррентные модели по точности и вычислительной сложности на длинных текстах, но их применение в условиях ограниченных ресурсов остается проблематичным. Иерархические гибридные модели обеспечивают более глубокую интеграцию методов, однако сложны в разработке и настройке, а их адаптивность к концептуальному дрейфу ограничена статичностью семантических связей.

Главное требование к архитектуре - адаптивность к концептуальному дрейфу, который в задаче квалификации текстов социальных медиа проявляется как появление новых терминов, сдвиг семантики существующих и изменение языковых паттернов с течением времени. Такая адаптивность обеспечивается через точечное применение ресурсоемких алгоритмов: они запускаются только для аномального контента, выявленного на ранних этапах обработки. Это позволяет обновлять онтологию и корректировать классификационные правила без переобучения всей системы. В отличие от классического ансамблирования, где классификаторы работают независимо, предлагаемая архитектура реализует итеративный обмен данными между нейросетевым и онтологическим модулями. Онтологическая спецификация при этом обеспечивает интерпретируемость решений, не снижая точности классификации.

2.3 Структура гибридного метода квалификации

На основе формальной модели квалификации (раздел 2.1) разработана трехуровневая архитектура гибридного метода. Архитектура реализует множественную принадлежность семантических единиц, использует семантические примитивы для построения признакового пространства и обеспечивает иерархическую обработку по схеме «признак → категория → сет».

В качестве языка моделирования процесса квалификации в исследованнии выбран объектно-ориентированный язык. Объектно-ориентированные языки моделирования появились в период с середины 70-х до конца 80-х годов, когда исследователи, поставленные перед необходимостью учитывать новые возможности объектно-ориентированных языков программирования и требования, предъявляемые все более сложными приложениями, вынуждены были начать разработку различных альтернативных подходов к анализу и проектированию. С 1989 по 1994 год число различных объектно-ориентированных методов возросло с десяти более чем до пятидесяти. Унифицированный язык моделирования (UML) является стандартным инструментом для создания «чертежей» программного обеспечения. Таким образом для построения архиектры программного решения

выбран UML, а конкретно: диаграмма пакетов, диаграмма классов,диаграмма состояний, диаграмма развертывания.

Общая схема процесса квалификации приведена на рисунке 1 в виде диаграммы деятельности.

Диаграмма, представляет собой сценарий выполнения процесса квалификации, а также отображение того, как процесс взаимодействует с другими процессами с точки зрения обмена сообщениями (информацией, документами и другими функциональными объектами).

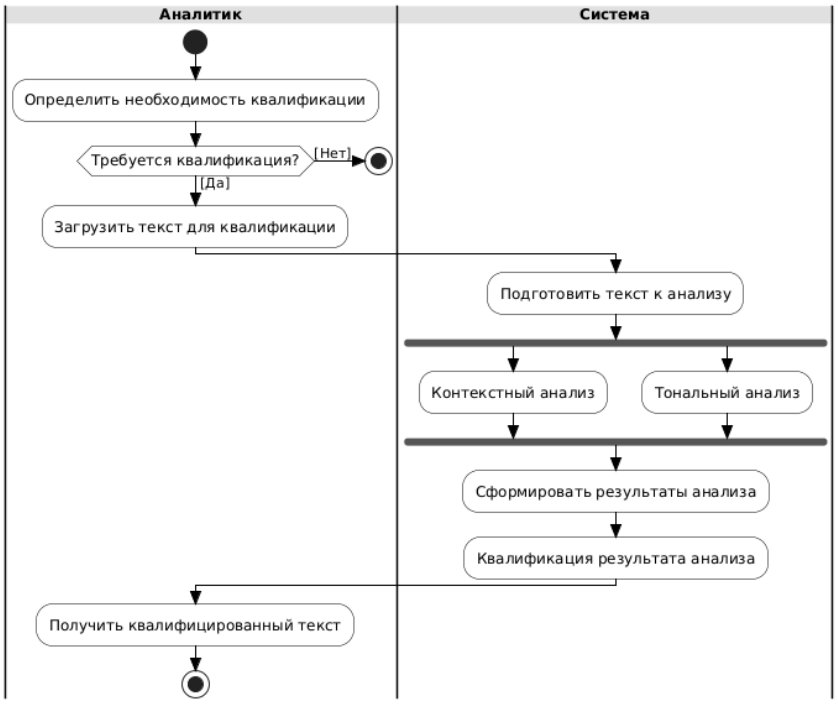


Рисунок 1 - Диаграмма деятельности процесса квалификации

После загрузки исходного текста выполняется его предварительная обработка: очистка и токенизация. Именно на этапе токенизации выделяются семантические примитивы - минимальные значимые единицы, которые в дальнейшем служат основой для построения признакового пространства. Далее данные поступают на второй уровень, где параллельно работают два модуля: контекстного и тонального анализа. Такая параллельная организация внутри уровня сокращает время обработки, не нарушая общей последовательной структуры пайплайна. Результаты обоих модулей объединяются на третьем уровне при итоговой квалификации.

Предлагаемая архитектура гибридного метода состоит из нескольких уровней (Рисунок 2).

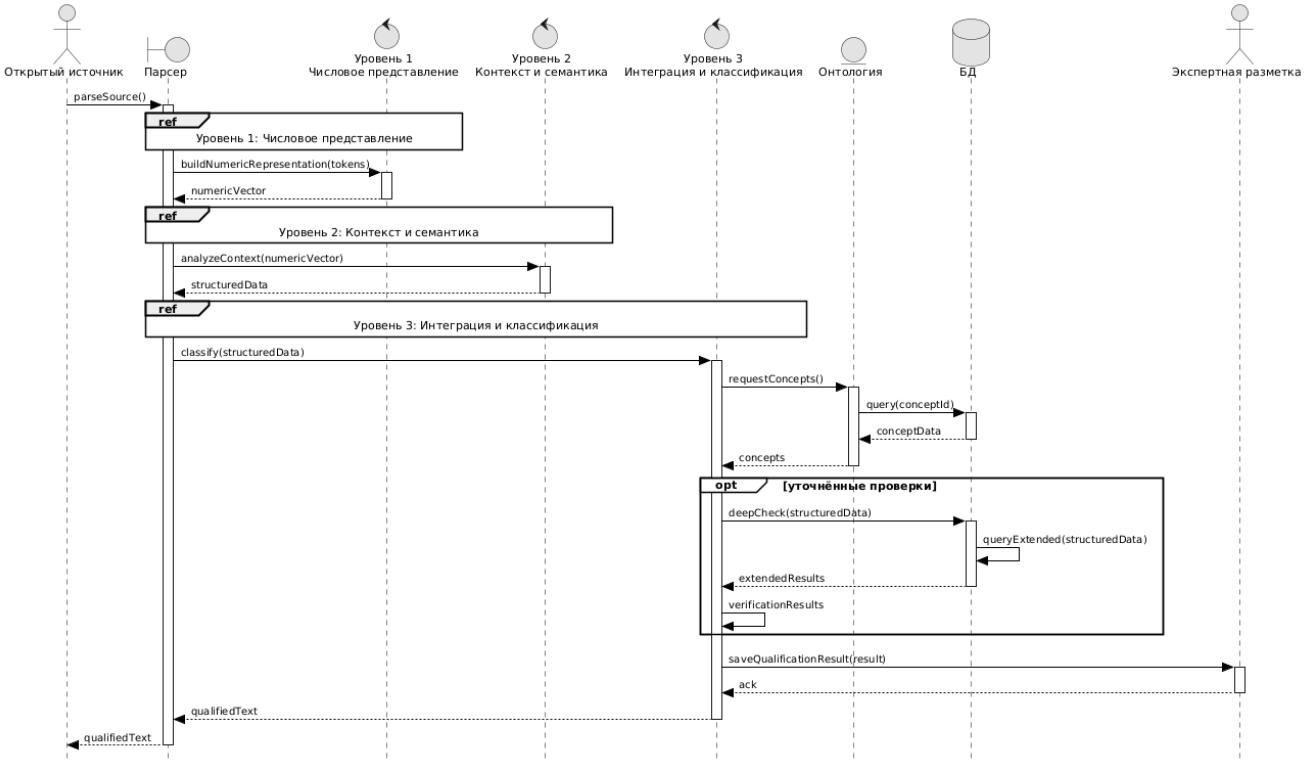


Рисунок 2 - Предлагаемая структура гибридного метода

На первом уровне исходный текст преобразуется в структурированное числовое представление. Формируется четыре типа данных: эмбеддинги токенов, частотные профили, тональную разметку и синтаксические деревья. Каждый из этих каналов захватывает определённый аспект содержания - от статистической значимости лексики до эмоциональной окраски.

Синтаксический анализ с помощью парсеров формирует деревья зависимостей, фиксирующие грамматическую структуру.

Частотный профиль строится с использованием векторных представлений. Размерность сокращается за счет использования BPE-токенизации и не влечёт значимых потерь для последующей классификации. Для тонального анализа используется сочетание словарного подхода и нейросетевого: маркеры классифицируем по валентности (положительной, отрицательной, нейтральной), по тексту вычисляем среднюю оценку и доли каждой тональности. Главное ограничение этого метода - он не учитывает взаимодействие маркеров в контексте.

Однако этот недостаток компенсируется на следующих уровнях, где в дело вступают эмбеддинги и семантический анализ.

Второй уровень добавляет контекст и смысловые связи между элементами.

Контекстный анализ основан на классификации Хомского:

—тип 0 - грамматики с фразовой структурой, или без ограничений. На структуру их правил не накладывается никаких ограничений. Это самый общий тип грамматик.Грамматики, которые относятся только к этому и не могут быть отнесены ни к какому другому типу, являются самыми сложными.

—тип 1 - Контекстно-зависимые (КЗ) и неукорачивающие грамматики. К этому типу относятся два основных класса грамматик. В КЗ-грамматиках при построении предложений заданного языка один и тот же нетерминальный символ может быть заменен различными терминальными цепочками в зависимости от контекста, в котором он встречается

—тип 2 - Контекстно-свободные (КС) грамматик. В правой части у них стоит всегда хотя бы один символ. Их отличие от предыдущих типов состоит в том, что левая часть правил должна состоять ровно из одного нетерминального символа. Такие грамматики еще называют неукорачивающими контекстно-свободными (НКС) грамматиками.

—тип 3 - Регулярные грамматики. В правой части правил грамматик этого типа может присутствовать не более одного нетерминального символа, причём он должен быть расположен во всех правилах одной грамматики с одной и той же стороны от цепочки терминалов, а требования к левой части правил совпадают с предыдущим типом. К этому типу относятся два эквивалентных класса грамматик: леволинейные и праволинейные (их название определяется местоположением нетерминального символа в правой части правил относительно терминальной цепочки). Для любой праволинейной грамматики можно построить эквивалентную ей леволинейную, задающую тот же язык, и наоборот.

Интернет-языки формально можно отнести к контекстно-зависимым (тип 1), так как он имеет множество контекстных ограничений. Естественный язык, обладая такими свойствами, как вложенность и перекрёстные зависимости, явно не

может быть охвачен грамматикой типа 3 или типа 2. Однако более точная классификация помещает его в промежуточный класс «слабо контекстно-зависимых» языков - подкласс типа 1, который, в отличие от полного типа 1, остаётся вычислительно разрешимым. Синтаксические формулы формализуют грамматические связи, задавая структуру, фиксируя зависимости и обеспечивая возможность автоматического анализа.

Семантическая близость вычисляется количественно - через косинусное сходство между семантическим пространством текста и базовым пространством обучающей выборки.

В качесетве стиломатерической метрики предлагается использовать дельту Берроуза. Классическая дельта Берроуза рассчитана на тексты от 1000 слов [14-17].

В настоящем исследовании используется модифицированная версия дельты Берроуза, адаптированная для коротких текстов. В отличие от классического подхода, оперирующего 100-200 наиболее частотными словами, предлагаемая модификация (формула 7) применяет частоты всех токенов BPE-словаря. Это позволяет сохранить статистическую значимость при малой длине текста: субсловное представление увеличивает эффективный объем выборки и, как следствие, повышает статистическую надежность частотных оценок, что согласуется с результатами [18, 19].

В отличие от классической дельты Берроуза, где она выступает самостоятельным классификатором, в предлагаемой гибридной модели этот метод используется как один из признаков в интегральном векторе (формула 9). Такое решение позволяет компенсировать ограничения стилометрического подхода за счёт других компонентов - контекстуализированных векторных представлений и тонального анализа. Принципиальное отличие от работ [20, 21], где стилометрические и семантические признаки объединялись на уровне решений (ансамблирование), заключается в том, что реализуется интеграция на уровне признакового пространства. Это позволяет учесть корреляции между стилем и тональностью уже на этапе обучения, а не только при финальном голосовании классификаторов.

Модифицированные версии стилометрических метрик показывают точность, сопоставимую с нейросетевыми подходами, при значительно меньших вычислительных затратах [22, 23]. Кроме того, комбинирование частотных характеристик с лексическими признаками позволяет эффективно анализировать тексты длиной до 50 слов [24].

Таким образом, использование модифицированной дельты Берроуза в гибридной модели квалификации является обоснованным. Адаптация метрики к коротким текстам обеспечивается за счет BPE-токенизации и включения в интегральный вектор признаков, а ее ограничения компенсируются другими компонентами гибридной архитектуры. Снижение точности на сообщениях длиной менее 10 слов отмечено как ограничение и вынесено в Заключение в качестве направления для дальнейших исследований.

Семантическая сеть аккумулирует результаты работы первых двух уровней, преобразуя их в структурированное представление знаний, которое передаётся на третий уровень для принятия итогового решения о категориальной принадлежности текста.

Третий уровень - интеграция и классификация: данный уровень предназначен для принятия итогового решения о категориальной принадлежности текста на основе всех извлечённых признаков и построенных семантических структур. Уровень обеспечивает интеграцию разнородной информации, полученной на предыдущих этапах, и её преобразование в классификационное решение.

В качестве интегрирующей архитектуры используются свёрточные нейронные сети (CNN), что позволяет обрабатывать входящие сообщения потоково, без накопления батчей. Входными данными для свёрточной сети служат не сырые тексты, а результаты их обработки на предшествующих уровнях: векторы частотных характеристик и эмбеддинги, синтаксические представления, параметры положения узлов в семантическом графе, показатель аномальности (дельта Берроуза). Все эти признаки приводятся к единой размерности через слой

линейной проекции и конкатенируются в вектор, который затем подаётся на свёрточные слои.

Каждому узлу графа, соответствующему уникальной лексеме, ставится в соответствие агрегированный вектор, включающий эмбеддинг слова, среднюю тональность по всем вхождениям в тексте и частотный вес. Таким образом, граф превращается в аннотированный граф признаков. Подача на CNN дополнительных характеристик, извлечённых из семантического графа (например, степень узла, близость к центру кластера), позволяет модели учитывать структурные свойства текста, выходящие за пределы локального контекста свёрточных фильтров. В свою очередь, графовые представления обогащаются частотными и тональными характеристиками, что даёт возможность учитывать не только качественные отношения между сущностями, но и количественные параметры их проявления в тексте.

Выходной слой нейросети формирует распределение вероятностей по категориям. В одноме точном режиме используется softmax-нормализация, в многмоеточном - независимые сигмоиды. Это позволяет тексту одновременно относиться к нескольким категориям с различными весами - решение соответствует принципу многопринадлежности, заложенному в формальной модели (раздел 2.1). Для коротких и зашумленных сообщений социальных медиа такая гибкость критична: один пост может одновременно касаться разных тем и нести несколько эмоциональных окрасок.

Онтологическая спецификация на верхнем уровне обеспечивает интерпретируемость решений. Результаты классификации соотносятся с понятиями и отношениями, заданными в онтологии. Для каждого решения формируется объяснение - набор признаков (онтологические отношения, графовые связи, ключевые маркеры), обосновывающих вывод. Противоречие аксиомам онтологии (системе ограничений) запускает углублённую проверку или экспертную разметку, что позволяет выявлять случаи концептуального дрейфа.

Накопление несоответствий между предсказаниями модели и онтологическими ограничениями служит индикатором дрейфа данных и

сигнализирует о необходимости обновления модели и онтологии. Полная обратная связь между уровнями потребовала бы переобучения всех компонентов в реальном времени, что вычислительно нереализуемо для потоковых данных. реализуется асинхронный механизм накопления ошибок с периодическим дообучением модели - он не обеспечивает мгновенной адаптации, но позволяет отслеживать изменения в структуре обрабатываемых данных и инициировать процедуры обновления.

Представленная трёхуровневая архитектура определяет структуру обработки данных и служит основой для программно-алгоритмической реализации. В частности, на первом уровне реализуются механизмы кэширования частотных характеристик, на втором - оптимизация графовых вычислений с асинхронным обновлением онтологии, на третьем - квантизация весов и пакетная обработка свёрточных сетей. Предложенная архитектура не предполагает обратных связей между уровнями в процессе обработки одного запроса, что обеспечивает детерминированность и предсказуемость вычислений при потоковой обработке данных.

2.4 Программно-алгоритмическая реализация и оценка эффективности

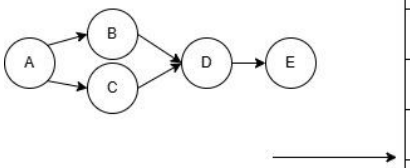
Задача квалификации потоковых данных из социальных медиа в реальном времени предъявляет высокие требования к производительности. Процесс квалификации включает два ресурсоемких этапа - контекстный и тональный анализ. Эти этапы могут выполняться параллельно, что сокращает общее время обработки. Однако при скорости поступления сообщений до тысяч единиц в секунду одного параллелизма на уровне задач недостаточно. Необходима комплексная оптимизация на всех уровнях вычислительной архитектуры.

В соответствии с трёхуровневой архитектурой распараллеливание реализуется на нескольких уровнях.

На верхнем уровне применяется декомпозиция задачи квалификации на функционально независимые подзадачи - контекстный и тональный анализ. Исходная комплексная задача разделяется на совокупность автономных подзадач, которые могут выполняться на различных вычислительных ядрах или узлах

кластера независимо друг от друга. Результаты обоих анализов синхронизируются перед этапом итоговой квалификации (уровень 3), что создаёт точку синхронизации в вычислительном графе.

Для формализации распараллеливания процесс квалификации представляется в виде ориентированного графа задач. Вершины соответствуют отдельным вычислительным операциям, рёбра - информационным зависимостям между ними. Исходная информация для распараллеливания может быть представлена в виде матрицы смежности, отражающей временные и логические зависимости между подзадачами (Рисунок 3).



	A	B	C	D	E
A	0	1	1	0	0
B	∞	0	∞	1	∞
C	∞	∞	0	1	∞
D	∞	∞	∞	0	1
E	∞	∞	∞	∞	0

Рисунок 3 - Исходная информация для распараллеливания процесса квалификации

Анализ такой матрицы показывает, что контекстный и тональный анализ (вершины B и C) не имеют взаимных информационных зависимостей на этапе выполнения. Это позволяет организовать их одновременный запуск. При этом результаты обоих анализов синхронизируются перед этапом итоговой квалификации (вершина D), что создаёт точку барьерной синхронизации в вычислительном графе.

Временной план выполнения построен по принципу критического пути с учетом выявленных зависимостей (Рисунок 4). Задачи 1 и 2 (контекстный и тональный анализ) запускаются одновременно на разных вычислительных ресурсах. Задачи 3, 4 и 5 (этапы постобработки и интеграции) выполняются последовательно после завершения всех параллельных ветвей и синхронизации результатов. Общее время обработки складывается из максимального времени

выполнения параллельных ветвей, времени на синхронизацию и финальной квалификации.

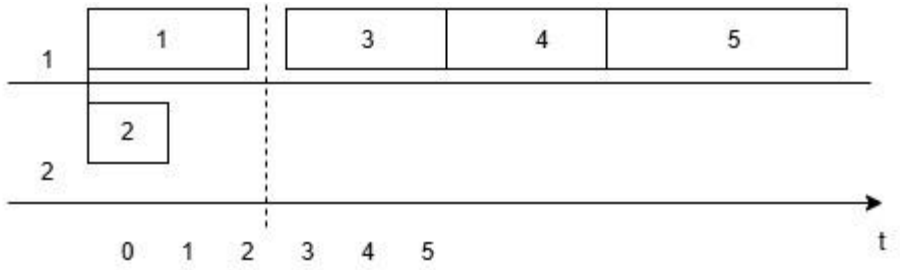


Рисунок 4 - Временной план распараллеливания процесса квалификации

Помимо распараллеливания задач, в архитектуре предусмотрена оптимизация на уровне алгоритмов.

Для первого уровня кэшируются частотные профили BPE-токенов; пороговое отсечение редких токенов (менее 3 вхождений в корпусе) сокращает размерность словаря без значимых потерь на валидации. На втором уровне кэшируется матрица расстояний между концептами в семантическом графе; она асинхронно перестраивается при обновлении онтологии, что исключает полный пересчёт графа при каждом запросе. Для типовых синтаксических конструкций предусмотрено кэширование шаблонов разбора. На третьем уровне свёрточные сети используют квантизацию весов и пакетную обработку. Сквозное кэширование промежуточных представлений (эмбеддингов, синтаксических шаблонов, итоговых решений для семантически близких сообщений) сокращает вычислительные затраты за счёт высокой повторяемости лексики в потоке социальных медиа.

Для обработки пиковых нагрузок архитектура предполагается горизонтальное масштабирование компонентов. Распределение сообщений между узлами одного типа выполняется согласованным хэшированием. Сообщения от одного автора или одной темы закрепляются за узлом для сохранения локальности кэша. Мониторинг загрузки управляет автоматическим запуском и остановкой дополнительных экземпляров.

Ключевая особенность архитектуры - механизм динамической адаптации маршрутизации потоков данных, основанный на результатах семантического

анализа и значении стилометрической метрики (дельты Берроуза). Механизм взаимодействия реализует распределение вычислительных ресурсов в зависимости от характеристик входного текста и не предполагает перестройки архитектуры при обработке одного запроса.

В предлагаемом решении дельта Берроуза выполняет две функции. Первая - количественная мера семантического сдвига. Дельта используется для управления эволюцией онтологических структур. При потоковой обработке она работает как детектор аномалий. Превышение порога сигнализирует о контенте, стилометрически отклоняющемся от типичных для категории образцов. Такой контент направляется на углубленный семантический анализ. Вторая - индикатор изменений в данных. Оценка величины семантического сдвига позволяет корректировать пороги с учетом текущего состояния онтологии.

На основе дельты Берроуза реализуется распределение вычислительных ресурсов: рядовой контент направляется по стандартному пути с использованием кэшированных результатов и статистических методов первого уровня; контент, идентифицированный как аномальный (с высоким значением дельты), поступает на приоритетную обработку с привлечением более ресурсоемких семантических и нейросетевых методов (уровни 2 и 3). Эти методы применяются точечно - только там, где это необходимо для качества квалификации. Такой подход обеспечивает баланс между глубиной анализа и пропускной способностью системы, что особенно важно при потоковой обработке больших данных из социальных медиа.

Разработанный механизм динамической адаптации, механизмы распараллеливания и кэширования составляют основу программно-алгоритмической реализации метода. Детальное описание программной библиотеки и оценка ее эффективности на реальных данных представлены в главе 5.

2.5 Выводы по главе 2

Проведенный во второй главе анализ показал, что ни один из существующих подходов в отдельности не является достаточным для решения задачи

квалификации динамичных потоков текстовых данных. Формальные онтологии обеспечивают строгий логический вывод и интерпретируемость, но не адаптируются к изменениям языка. Статистические методы хорошо распознают частотные паттерны, но теряют контекст и семантические связи. Нейросетевые архитектуры показывают высокую точность на статических тестах, но остаются непрозрачными и требуют больших вычислительных ресурсов. Следовательно, необходима гибридизация - комбинация методов в единой архитектуре, компенсирующая ограничения каждого из них.

1. Разработана формальная модель квалификации неструктурированных текстовых данных, задаваемая кортежем $M = \langle W, C, R, \Omega, \Phi \rangle$. Модель реализует механизм многопринадлежности через вектор вероятностей и трёхуровневую структуру «признак → категория → сет», что обеспечивает учёт контекстной зависимости и полисемии при обработке коротких зашумлённых сообщений и создаёт теоретическую основу для алгоритмической и программной реализации в последующих разделах.

2. Сравнительный анализ существующих архитектур гибридных систем квалификации текстов показал, что конвейерные системы накапливают ошибки и не обеспечивают интерпретируемости, ансамблевые (в том числе на LLM) - ресурсоёмки и неприменимы для потоковой обработки в реальном времени, иерархические - сложны в настройке и ограниченно адаптивны к концептуальному дрейфу. Главное требование к архитектуре - адаптивность к концептуальному дрейфу, обеспечиваемая точечным применением ресурсоёмких алгоритмов только к аномальному контенту, что позволяет обновлять онтологию и корректировать классификационные правила без переобучения всей системы.

3. Разработана трёхуровневая архитектура гибридного метода квалификации: первый уровень (признаковый анализ) извлекает количественные характеристики; второй уровень (семантический анализ) вводит контекст, строит семантическую сеть и вычисляет дельту Берроуза как показатель аномальности; третий уровень (интеграция и классификация) принимает итоговое решение на основе свёрточной сети и сопоставляет результат с онтологическими классами.

4. Разработан механизм распараллеливания контекстного и тонального анализа и многоуровневого кэширования (частотных характеристик, графовых структур, промежуточных результатов CNN). Указанные решения составляют основу программно-алгоритмической реализации метода.

Разработанные модель, структура и алгоритмические решения служат основой для программной реализации метода квалификации, представленной в главе 3. В главе 3 математический аппарат дополняется формализацией всех этапов обработки, в главе 4 описывается внедрение разработанного модуля в действующую парсинговую систему и оценивается его эффективность на реальных и синтетических данных.

3. РАЗРАБОТКА МЕТОДА СЕМАНТИЧЕСКОГО АНАЛИЗА ДАННЫХ

3.1 Семантический анализ: понятие, функции, применение в средствах автоматизированной обработки данных

Н. Хомский сначала формулировал лингвистическую концепцию «трансформационной порождающей грамматики». Её конструирование осуществлялось согласно дедуктивно-аксиоматическим принципам развёртывания логических исчислений. Благодаря динамическому переосмыслению важнейшим уровнем грамматической модели стала грамматика фразовых структур, которая в свой состав включала контекстно самостоятельные правила развёртывания, способные быть неоднократно использованными по отношению к исходным символам, что в итоге вырабатывало терминальную цепочку, называемую также ядерным предложением.

Первая версия вызвала в научном сообществе оживлённые дискуссии, которые в конечном итоге привели к появлению новых версий. В 1965 году Н. Хомский опубликовал свою новую работу - «Аспекты теории синтаксиса», которая в большей степени известна в генеративном языкознании как Стандартная теория. В соответствии с ней грамматика включает в себя следующие компоненты:

—синтаксический компонент - представлен трансформационными правилами, которые преобразуют глубинные структуры в поверхностные, и

правилами лексикона с правилами порождения фразовых структур, которые совместно обеспечивают порождение глубинных структур, выполняющих роль носителей исходной, семантически релевантной информации;

—семантический компонент - осуществляет семантическую интерпретацию глубинных структур;

—фонологический компонент - осуществляет фонетическую интерпретацию поверхностных структур.

Семантический анализ это этап в обработке естественного языка (NLP), направленный на извлечение, интерпретацию и структурирование смысла текста: значений слов, фраз, предложений и их взаимосвязей в контексте. Множество функций семантического анализа включает в себя: информационную, описательную, структурную, преобразовательную.

Информационный аспект отвечает за раскрытие сути анализируемых материалов. Позволяет определить основную мысль текста, выявить авторскую позицию, понять контекст и смысловую нагрузку представленных данных. Описательный компонент необходим для фиксации структуры, лексических особенностей и синтаксических конструкций.

Структурный анализ необходим для выявления причинно-следственных связей и определения последовательности изложения. Преобразовательная функция обеспечивает обработку входного массива данных для последующего анализа.

На рисунке 5 представлена диаграмма деятельности процесса семантического анализа.

Процесс включает в себя этап сбора информации, предварительную обработку, определение семантических ролей, выявление связей между семантическими единицами, обработку результатов.

Сбор информации, как правило, выполняется в автоматизированном режиме, с применением специальных средств: парсеров.

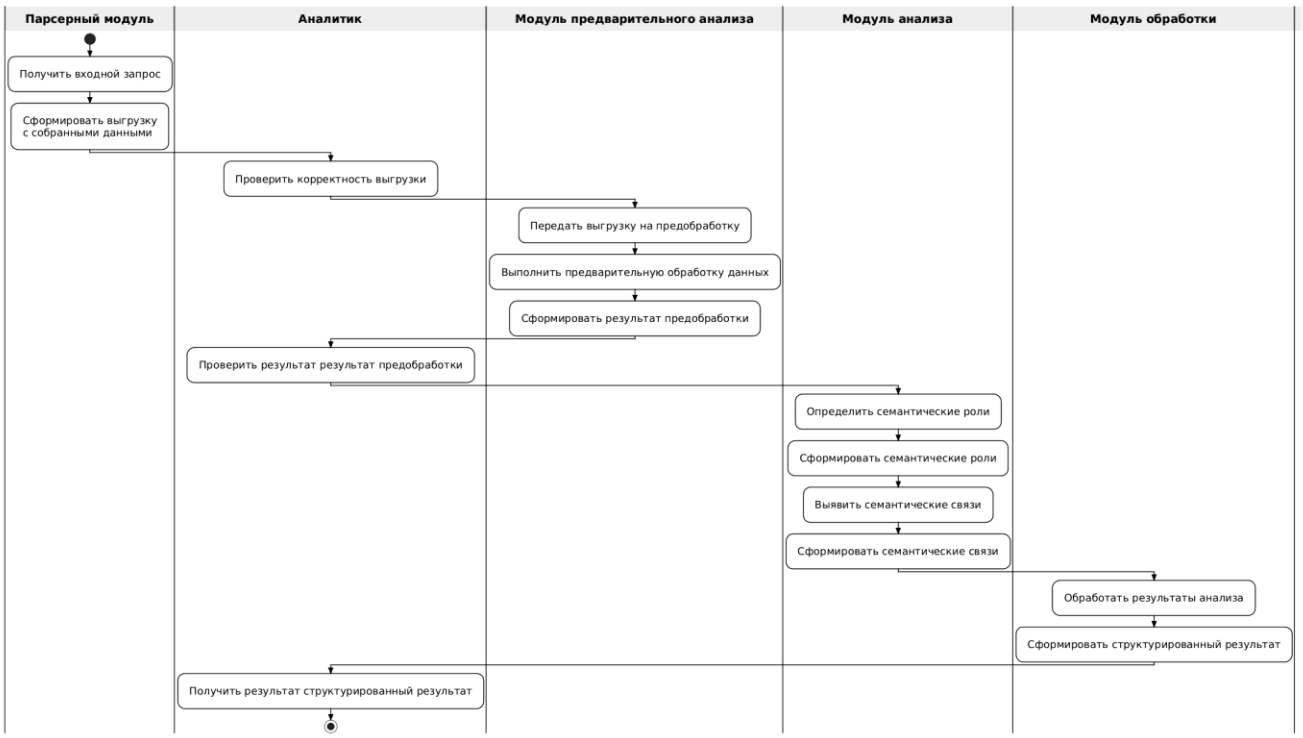


Рисунок 5 - Диаграмма деятельности

Парсеры позволяют собирать целевой контент Интернет - ресурсов, отталкиваясь исключительно от запроса. Тип инструмента определяется входными параметрами, а именно источником данных. Как правило выделяют 3 вида парсинга:

—веб-парсинг -работа с HTML-кодом веб-страниц;

—API-парсинг -получение данных через программные интерфейсы;

—документо-ориентированный парсинг -обработка текстовых файлов различных форматов.

Следующий этап предварительной обработки выполняет аналитик. Для этого он использует модули предварительного анализа.

Определение семантических ролей позволяет глубже понять структуру текста и связи между его элементами. Семантические роли определяют так: устанавливают, как участник ситуации связан с действием, которое обозначает глагол (предикат). При анализе социальных медиа это помогает выяснить, кто автор текста и кому он адресован. В некоторых случаях такой анализ позволяет найти самых вовлечённых пользователей, а также локации, связанные с описанными действиями, если эти локации как-либо обозначены.

В современной лингвистике используют три подхода к семантическим ролям: предикатно-специфический: для каждого глагола создают уникальный набор ролей; обобщённый: обходятся базовыми ролями; классический: применяют стандартный набор ролей.

Выявление семантических связей позволяет обнаружить и проанализировать смысловые отношения между элементами текста, устанавливающее контекст и взаимосвязи между синтаксическими единицами.

Современные подходы к обнаружению таких связей включают векторные представления слов, в которых близость векторов отражает смысловую близость; контекстуальные эмбеддинги, учитывающие контекст при определении значений; матричные методы, анализирующие взаимосвязи посредством матриц термин-документ; наконец, онтологические подходы, опирающиеся на заранее определенные словари и тезаурусы.

Результаты анализа представляются конечным пользователям в структурированном виде.

Рассматривая подробнее процесс предобработки данных, важно рассмотреть процесс токенизации. Это процесс разделения текстовых данных на понятные модели машинного обучения токены. Токен может представлять собой целое слово, часть слова, либо символ.

Процесс токенизации заключается в решении четырех задач (Рисунок 6):

— очистка текста от лишних символов - процесс удаления спецсимволов, артефактов, эмодзи, лишних пробелов;

— разбиение на потенциальные токены (сегментация) - процесс разделения текстовых данных на значимые элементы;

— нормализация - процесс приведения токенов к единой корневой форме;

— индексация - процесс присвоения уникальных идентификаторов токенам для.

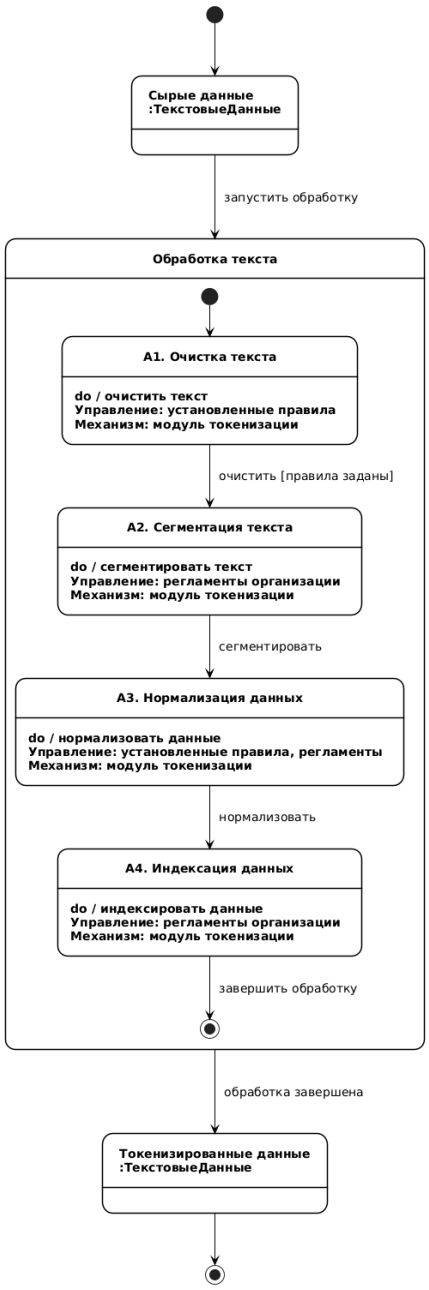


Рисунок 6 - Процесс токенизации

Расширяя вышеописанный процесс, можно говорить о том, что: исходя из результата процесса предобработки переход к процессу выявления связей между семантическими единицами и анализа смысловых отношений должен происходить исходя из вопроса разделения контента на подходящий или шумовой, относительно задачи аналитика. Подходящий контент, также может быть разбит по микротемам обозначенных задач. Таким образом процесс предобработки, подразумевающий в том числе разбиение набора текстовых данных на токены, и процесс определения соответствия данных позволяют формировать наборы данных для тональных и семантических словарей, необходимых для процесса квалификации контента.

Также данный процесс (Рисунок 7). можно определить, как процесс, предшествующий процессу основной квалификации данных.



Рисунок 7 - Упрощенный алгоритм токенизации

Семантический анализ выполняет информационную, описательную, структурную и преобразовательную функции, а его процесс включает сбор данных, предварительную обработку, определение семантических ролей, выявление связей и обработку результатов. Процесс токенизации подготавливает данные для последующей обработки. Вместе с тем эффективность семантического анализа зависит от качества предобработки и выбора подхода к семантическим ролям, а разделение контента на целевой и шумовой требует дополнительных критериев.

3.2 Разработка алгоритма для обработки данных:

3.2.1 Базовый алгоритм BPE применяемый для обработки текстовых данных

В настоящее время большинство парсинговых и аналитических систем работают на основе процесса токенизации, так как целевое извлечение данных экономит ресурсы.

Однако при работе с социальными медиа целевое извлечение данных не является рациональным подходом для парсинга. Большая часть контента в соцмедиа доступна только авторизованным пользователям. Чтобы его парсить необходимо поддерживать сессию и эмулировать работу пользователя, при этому рискуя заблокировать аккаунт. Каждая социальная медиа использует собственные алгоритмы для определения бот-систем, используемых для сбора данных. Стандартными методами защиты являются сканирование цифрового отпечатка, CAPTCHA и аналоги, а также отслеживание активности запросов. Но практически все социальные медиа предоставляют доступ по API (Application Programming Interface) - интерфейс, позволяющий получать данные из внешних источников в структурированном виде. Это значительно упрощает задачу парсинга, так как имитация действия пользователя в этом случае служит защитой от информации распространяемой соцсетью, в целях защиты от парсинга, и позволяет безболезненно собирать интересующий контент. При таком подходе создается практически копия источников информации, и квалификация получаемого контента становится не просто базой для дальнейшего анализа, а необходимостью для создания релевантной выборки. При этом обработка больших текстовых данных с применением LLM-моделей становится неэффективной. Так как методология процесса квалификации требует интеграции в свою структуру комплексного подхода к лексическому анализу с применением лексикографических инструментов. Также на обработку влияет массив контента на разных языках, что усложняет создания разного рода словарей. С точки зрения машинного обучение модернизация процесса квалификации может быт решена переходом на модели, которые обрабатывают текст на уровне байтов - Byte-level модели. В отличие от традиционных языковых моделей, которые обычно работают на уровне слов или частей слов, byte-level модели эффективно обрабатывают различные языки и сценарии, в том числе со сложной грамматикой и морфологией, на уровне байтов.

Выделяют 3 основных вида byte-level моделей:

— простые байтовые модели (Pure Byte-Level): Работают с последовательностью байтов напрямую, без построения промежуточного словаря.

— модели на основе Byte-Level BPE: Используют адаптацию BPE для байтов для создания эффективного словаря.

— модели со специальными байтовыми архитектурами: Используют уникальные архитектурные решения для эффективного моделирования длинных байтовых последовательностей.

Преимущество такого подхода в работе напрямую с 256 возможными байтами. Несмотря на побайтовую обработку, подход не является масштабно ресурсоемким за счет использования Byte-Pair Encoding (алгоритм BPE), который также используется в GPT и BERT, разница в том, что в случае byte-level моделей, алгоритм применятся к байтам (Рисунок 8).

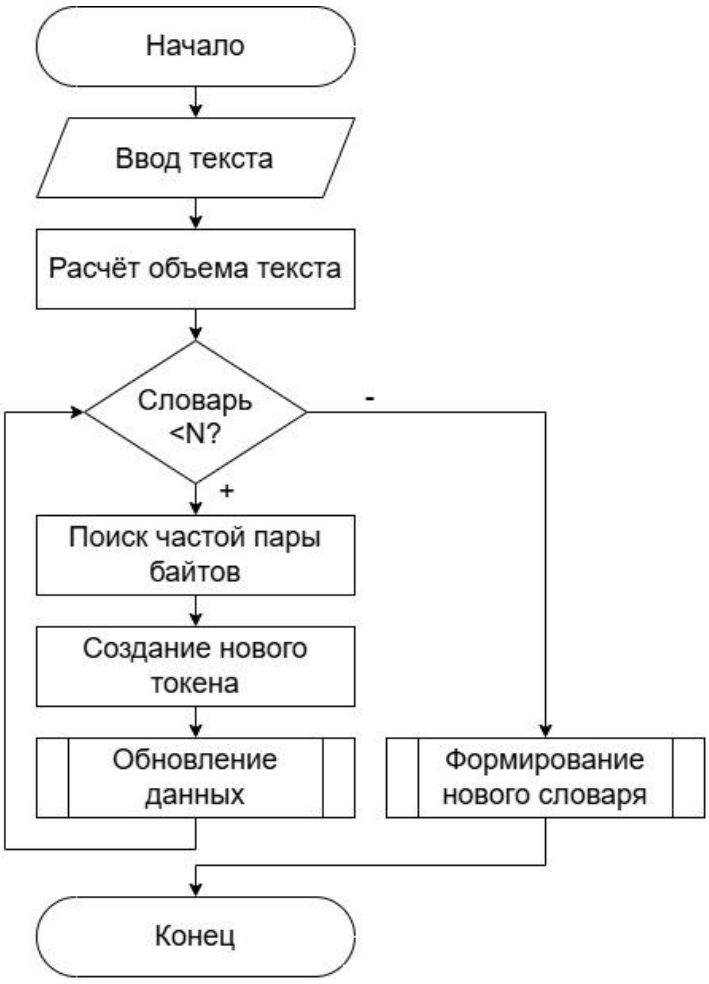


Рисунок 8 - Алгоритм построения словаря с применением BPE

Процесс построения словаря BPE можно описать следующей последовательностью шагов:

1. Инициализация словаря: На вход алгоритму подается обучающий корпус текстов. Начальный словарь формируется как множество всех уникальных байтов (или символов) в корпусе, где каждому элементу присваивается частота его встречаемости.

2. Алгоритм входит в цикл, на каждой итерации которого выполняет следующие действия:

— поиск наиболее частой пары: Производится поиск самой частотной последовательной пары байтов (например, пары, соответствующих символам «a» и «b» в кодировке UTF-8) во всем корпусе;

— создание нового токена: Найденная пара объединяется в новый единый токен;

— обновление данных: Словарь пополняется новым токеном. Все вхождения этой пары в тексте корпуса заменяются на вновь созданный токен. Частоты встречаемости всех возможных пар пересчитываются с учетом произведенных замен.

3. Цикл повторяется до достижения целевого размера словаря, заданного заранее как гиперпараметр. В результате формируется компактный и эффективный словарь токенов, оптимально подстроенный под конкретный набор данных.

Одним из весомых преимуществ таких моделей является кросс-лингвистичность и отсутствие проблем с OOV (Out-of-Vocabulary). Так как модель изначально обучена работать с мельчайшими единицами, также не возникает проблем с опечатками и условностями цифрового жаргона. В совокупности эти преимущества повышают надежность обработки множества неструктурированных данных, полученных парсерами из сети Интернет.

Рассмотренный алгоритм BPE-токенизации обеспечивает эффективное разбиение текста на подсловные единицы и решает проблему редких и неизвестных слов, а byte-level модели дополнительно обеспечивают кросс-лингвистичность и устойчивость к лингвистическому шуму. Вместе с тем BPE не учитывает семантику

и контекст: токены формируются по частотному критерию, а не по смыслу, что ограничивает его применение в задачах, требующих понимания смысловых связей. Это обосновывает необходимость дополнения BPE алгоритмами семантического и тонального анализа.

3.2.2 Базовые алгоритмы семантического и тонального анализа данных

Обработка семантических связей (Semantic Relation Processing) - это комплексная область компьютерной лингвистики и обработки естественного языка, которая занимается выявлением, анализом, формализацией и использованием смысловых отношений между элементами языка в тексте.

Сам процесс обработки семантических связей состоит из нескольких ключевых фаз, которые могут повторяться и уточняться (Рисунок 99). Его можно визуализировать как конвейер, на выходе которого получается семантический граф или набор фактов, где узлы - это сущности и концепции, а ребра - это связи между ними.

Первый этап предобработки текста: токенизация и нормализация. На следующем этапе определяются части речи и каждому токену присваивается грамматическая категория, исходя из категории выявляются зависимости. Определяются грамматические связи: подлежащее, сказуемое, дополнение, определение. Далее на основе синтаксической структуры извлекаются смысловые отношения, в том числе определяются смысловые роли: агент, тема, источник. На основе всех полученных данных формируется граф знаний. В котором узлы - сущности и концепции, ребра - связи. К графу применяются этапы проверки истинности. В дальнейшем дерево служит для выявления смысловых отношений и определения тональности.

Определение тональности или тональный анализ - процесс, с помощью которого возможно определить эмоциональную окраску, выраженную в текстовых данных.



Рисунок 9 - Алгоритм процесса обработки семантических связей

С точки зрения лингвистики и разработки задача тонального анализа достаточно сложная, так как необходимо переложить принципы определения субъективности, сарказма, иронии, сравнения, отрицания, позитивной и негативной оценки, а также культурные и отраслевые особенности.

В основе этого процесса также лежит предобратка текстовых данных (токенизации, нормализация и лемматизация), далее частотный анализ, оценка зависимостей на основе контекстуальных и семантических зависимостей морфем, процедура векторизации.

На основе множества графов семантических связей и результатов тонального анализа формируется семантическое пространство.

Построение семантического пространства - это процесс создания математической модели, в которой языковые единицы представлены в виде векторов (точек) в многомерном пространстве таким образом, что геометрическое расстояние между этими векторами отражает их семантическую близость. Чем ближе два вектора в этом пространстве, тем более похожими по смыслу являются соответствующие им слова.

В основе процесса лежит гипотеза распределения семантики и векторное представление смысла. Основная идея гипотезы распределения семантики заключается в том, что семантическое значение языковой единицы определяется множеством контекстов, в которых она появляется. Процесс определения множества контекстов называется дистрибуцией. Слова с похожим распредлением имеют похожее значение, с разным - разное значение. Гипотеза служит основой для построения векторных семантических моделей. С практической точки зрения (Рисунок10) семантическое пространство формируется из текстового корпуса, в случае рассматриваемых процессов из общего банка данных парсинга, далее производится подсчет совстречаемости (появление нескольких единиц текста вместе в одном контексте) и на основе этого формируется векторное представление каждого слова, заключительным этапом производится расчет семантической близости, например, на основе рассматриваемой ранее модели «Word2Vec» и принципа «мешка слов».

Если формирование корпуса простой процесс с точки зрения парсинга, то следующий этап подсчета встречаемости становится более ресурснозатратным. Задается «Окно встречаемости» - количество слов слева и справа от целевого значения, строится матрица встречаемости, в которой строки и столбцы - это все уникальные слова из результатов парсинга. На пересечении строки i и столбца j стоит число, показывающее, сколько раз слово j встречалось в окрестности (в контексте) слова i. Так как при таком подходе возникает большое количество шума внутри матрицы, то размерность возможно сократить. Как раз на данном этапе применяются модели векторизации, позволяющая напрямую оптимизировать

положение векторов так, чтобы вероятность встретить соседние слова была максимальной.



Рисунок 10 - Свернутый алгоритм построения семантического пространства

С точки зрения квалификации контента используется совокупность всех вышеописанных методов.

Таким образом, рассмотренные алгоритмы семантического и тонального анализа обеспечивают извлечение смысловых отношений, построение графа знаний и определение эмоциональной окраски текста, однако каждый из них в отдельности имеет ограничения: семантический анализ требует ресурсоёмкой предобработки и зависит от качества грамматических парсеров, а тональный анализ чувствителен к сарказму, иронии и контекстуальной многозначности. Семантическое пространство, формируемое через совстречаемость и векторизацию, позволяет частично компенсировать эти ограничения за счёт учёта дистрибуции слов, но не решает задачу квалификации в полном объёме.

3.3 Алгоритм квалификации неструктурированных данных

3.3.2 Базовое представление алгоритма

Исходя из базовых алгоритмов предобработки текстовых данных, представленных в разделе 3.2, перед тем как квалифицировать информацию, необходимо выполнить:

— токенизацию;

— получить эмбединги;

— выяснить контекст;

— рассчитать дельту Берроуза, необходимую для определения возможности вброса информации;

— построить семантическое пространство.

При этом принципиальная невозможность однозначной квалификации сетевого контента является следствием его фундаментальных свойств как объекта исследования. С точки зрения обработки неструктурированных интернет-данных аналитик сталкивается с динамическими семиотическими системами, чье значение конструируется в реальном времени.

Цифровой контент существует в условиях размытого и динамичного контекста. На интерпретацию текста, изображения или видео влияют комментарии пользователей, связанные публикации и алгоритмическая выдача платформы. Субъективность восприятия дополнительно усиливается индивидуальным опытом потребителя. Значения языковых единиц в интернете не фиксированы, а формируются внутри конкретных субкультур и постоянно переопределяются при столкновении различных конвенций. В правовой сфере отсутствие унифицированных международных норм ведет к тому, что один и тот же контент может оцениваться по-разному в зависимости от юрисдикции.

Таким образом, проблема однозначной квалификации является не операционной, а методологической - она вытекает из самой природы интернет-контента как многомерного, контекстуально-зависимого и постоянно эволюционирующего объекта.

Процедура квалификации в общем смысле является присвоением каких - либо категорий данным. С точки зрения исследуемого процесса, связанного с потоком неструктурированной информации квалификация является присвоением признаков контенту, когда сочетание признаков рождает категорию, сочетание которых является сетом. Присвоение степени опасности или степени заинтересованности клиента в контенте является связанной, но самостоятельной задачей. На рисунке 11 дана обощенная концептуальная схема квалификации неструктурированного контента.

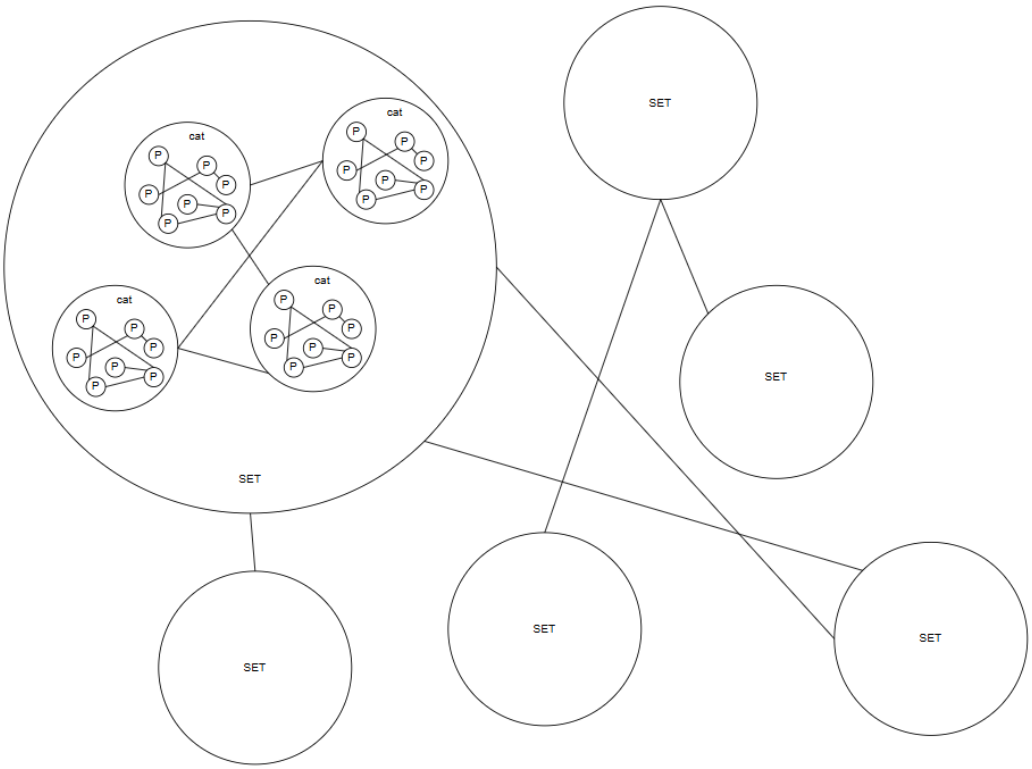


Рисунок 11 - Концептуальное представление квалификации

Общая концепция представляет собой вложенный граф, так как каждый узел содержит внутри себя другие графы. Такой тип графа позволяет скрывать ненужные детали на верхних уровнях и оперировать целыми группами объектов (подграфами) как единым узлом, что становится важным на этапе аналитических заключений и предшествующего ему анализа данных.

Для предложенной гибридной модели квалификации текстов оптимальным подходом является использование контекстуализированных эмбеддингов, получаемых с помощью моделей семейства BERT. В отличие от статических

моделей (Word2Vec, GloVe), которые присваивают слову единственный фиксированный вектор независимо от контекста, BERT генерирует динамические представления: одно и то же слово получает различные векторы в зависимости от смыслового окружения. Исследования показывают, что BERT устойчиво превосходит статические модели в задачах, требующих тонкого семантического различения.

Модели на основе трансформеров требуют больших вычислительных ресурсов, чем статические методы (Word2Vec), однако для задачи квалификации достаточно энкодерных моделей с уменьшенной размерностью. При этом BERT-модели имеют ограничения. Они чувствительны к иронии и сарказму, что для социальных медиа является существенной проблемой; в предлагаемом решении этот недостаток частично компенсируется на втором уровне за счёт синтаксического анализа. Кроме того, стандартное ограничение по длине последовательности (512 токенов) не является критичным для коротких сообщений, составляющих основной объём входного потока.

Для социальных медиа существенна субсловная токенизация (WordPiece), используемая в моделях BERT. Она позволяет обрабатывать редкие слова, неологизмы и орфографические ошибки, характерные для интернет-коммуникаций, что даёт преимущество перед методами, работающими на уровне целых слов.

Таким образом, для гибридной модели квалификации выбран DistilRuBERT. Время инференса составляет 45 мс на сообщение, что удовлетворяет требованиям потоковой обработки. Базовая конфигурация (Конфигурация 1) использует DistilRuBERT (768) и CNN-классификатор. В экспериментальной части (глава 5) для сравнения вычислительной эффективности использована облегчённая конфигурация (Конфигурация 2) на основе paraphrase-multilingual-MiniLM-L12-v2 (384) и Random Forest. Обе конфигурации реализуют одну формальную модель (раздел 2.1), различие заключатеся в типе энкодера и классификатора.

3.3.3 Математическое представление алгоритма

Категоризация текста выполняется на основе трёх аспектов: семантического, синтаксического и тонального. Каждый аспект формирует соответствующий вектор признаков; их совокупность определяет принадлежность текста к категории. Ранее были описаны три уровня анализа, на которых извлекаются эти признаки. Настоящий раздел детализирует механизм перехода от уровня отдельных признаков к уровню контекстных зависимостей, представленных в виде множеств.

В данном случае каждое понятие рассматривается как пространство с множеством определений, что позволяет учитывать синонимию и полисемию, характерные для неструктурированных текстовых данных. Синонимия (разные лексические единицы, выражающие одно понятие) и многозначность (одна лексическая единица, выражающая разные понятия) создают трудности при построении систем автоматической классификации, так как требуют учета контекстуальной вариативности языковых средств.

Исходный текст подвергается процедуре токенизации с использованием алгоритма Byte Pair Encoding (BPE), который обеспечивает эффективное разбиение текста на подсловные единицы с учетом частотности их встречаемости. В результате токенизации формируется последовательность идентификаторов токенов из предопределенного словаря.

Для преобразования текста в последовательность чисел, понятную модели, применяется алгоритм BPE-токенизации (формула 1):

$$T = T_{BPE}(S) = [t_1, t_2 \ldots t_n], t_i \in \{1, 2 \ldots |V|\}, \quad (1),$$

где:

$T_{BPE}(S)$ - токенизация текстового массива S на основе алгоритма BPE;

V - словарь;

t_i - идентификатор токена в словаре;

V - словарь токенов;

n - длина последовательности токенов.

Данный алгоритм был предложен в 2016 году в работе Рико Сенриха и соавторов «Нейронный машинный перевод редких слов с использованием подслов» [18]. Алгоритм байт-парного кодирования (BPE) изначально разработан для сжатия данных. В NLP адаптирован для решения проблемы редких и неизвестных слов путём разбиения слов на подслова. Стал стандартным методом токенизации в моделях GPT и BERT.

Каждому токену сопоставляется плотный вектор фиксированной размерности - эмбеддинг. В векторном пространстве семантически близкие токены располагаются рядом Последовательность токенов преобразуется в матрицу эмбеддингов извлечением соответствующих строк из матрицы подстановки. (Формула 2) [10]. Идея представления слов в виде плотных векторов (эмбеддингов) восходит к нейронной вероятностной языковой модели. Word2vec популяризировал этот подход. GloVe предложил альтернативный метод, основанный на глобальной статистике .

$$X = E[T] \in R^{n \times d}, \quad (2),$$

где:

T - токенизированный текст;

E - матрица эмбеддингов размерности [|V| × d];

d - размерность эмбеддингового пространства;

X - матрица эмбеддингов токенов размерности n×d.

Извлечение синтаксической характеристики представлено формулой 3. Работа Кима показала, что простая архитектура CNN с предварительно обученными эмбеддингами достигает высоких результатов в классификации предложений. CNN обеспечивают высокую скорость инференса и низкую задержку, что критично для систем реального времени [26,85]:

$$H_{syn} = CNN_{syn}(X) \in R^{h_{syn}}, \quad (3),$$

где:

H_{syn} - вектор синтаксических признаков размерности h_{syn};

h_{syn} - размерность пространства синтаксических признаков;

$CNN_{syn}(X)$ - сверточная нейронная сеть для анализа синтаксиса.

Семантические характеристики извлекаются методом латентного размещения Дирихле, представлено формулой 4 [86]:

$$H_{sem} = LDA(S) \in R^{h_{sem}} \quad , \tag{4}$$

где:

H_{sem} - вектор семантических признаков размерности h_{sem};

$LDA(S)$ - функция латентного размещения Дирихле, примененная к текстовому корпусу S.

Для определения средней лексической тональности используется формула 5 [87]:

$$b_{ton} = \frac{1}{N}\sum_{i=1}^{N} mood(t_i), b_{ton} \in R \tag{5}$$

где:

b_{ton} - средняя лексическая тональность текста;

$mood(t_i)$ - оценка тональности i-го токена (-1, 0, +1);

N - длина последовательности.

Для определения общей оценки тональности используется формула 6. Softmax - стандартная функция нормализации для многоклассовой классификации. Она преобразует вектор исходных чисел, часто называемых логитами, в вектор вероятностей. В результате все выходные значения становятся положительными и в сумме дают ровно единицу, то есть образуют корректное распределение вероятностей. Многозадачное обучение позволяет совместно использовать представления между задачами [88, 89]:

$$H_{ton} = soft\max(W_{ton} * H_{sem} + b_{ton}), \tag{6}$$

где:

H_{ton} - вектор вероятностей тональности (позитив/нейтрально/негатив);

H_{sem} - семантическая характеристика;

W_{ton} - обучаемая матрица весов размерности;

$b_{ton} \in R$ - вектор смещений размерности числа классов тональности.

Параллельно производится количественная оценка степени аномальности анализируемого текста относительно эталонных текстов, принадлежащих к определенному сету. В качестве меры аномальности используется дельта Берроуза - метрика, изначально разработанная для задач стилометрии и проверки авторства, но адаптированная в настоящем исследовании для выявления потенциальных вбросов в рамках категории (формула 7) [14]:

$$\Delta_B = \frac{1}{|V_{pr}|} \sum_{i \in V_{pr}}^{N} \left| \frac{f_i - \mu_i}{\sigma_i} \right|, \quad (7),$$

где:

Δ_B - дельта Берроуза;

V_{pr} - множество токенов BPE-словаря, присутствующих в анализируемом тексте;

$|V_{pr}|$ - число присутствующих токенов;

μ_i - средняя частота i-го токена в стандартных текстах;

f_i - частота i-го токена в анализируемом массиве;

σ_i - стандартное отклонение i-го токена.

Для использования аномальности в последующих вычислениях её необходимо нормировать [0, 1] с помощью сигмоидной функции и обучаемого параметра масштаба. Сырая аномальность нормируется в диапазон [0, 1] посредством сигмоидного преобразования с обучаемым параметром масштаба (формула 8) [90]:

$$\delta = \sigma\left(\frac{\Delta_B}{\gamma}\right), \sigma(x) = \frac{1}{1 + e^{-x}}, \quad (8),$$

где:

Δ_B - дельта Берроуза;

γ - обучаемый параметр масштаба;

δ - нормированная аномальность в диапазоне [0, 1].

Косинусное сходство - стандартная мера близости векторов в информационном поиске [28]. Механизм внимания позволяет модели фокусироваться на важных частях текста. Применение косинусного расстояния между частями текста для детекции сарказма является новым подходом. Чем выше

D_{parts}, тем сильнее различаются начало и конец текста. Саркастические высказывания часто строятся на контрасте: позитивное начало и негативная концовка (или наоборот). Этот признак улавливает динамику тональности внутри текста. (формула 9).

$$D_{parts} = 1 - \cos(Emb_1, ...Emb_n) \in [0, 2], \quad (9),$$

где:

$Emb_1 = \frac{1}{k}\sum_{i=1}^{k} X_i$ - усреднённый эмбеддинг первых k токенов (обычно k=⌊n/3⌋);

$Emb_n = \frac{1}{k}\sum_{i=n-k+1}^{n} X_i$ - усреднённый эмбеддинг последних k токенов;

n - длина последовательности;

X_i - эмбеддинг i-го токена (формула 2);

$\cos(u, v) = \frac{u * v}{\| u \| * \| v \|}$ - косинусное сходство.

В отличие от формулы (9), основанной на словарном подсчёте, формулы 10 - 13 позволяют вычислять признак семантическое противоречие между темой текста (именными группами) и контекстом (остальными словами). Метод основан на анализе семантического диссонанса [91].

Используется разметка на части речи (POS-tagging) для выделения именных групп (формула 10):

$$NP(S) = \{np_1, np_2, ..., np_m\}, \quad (10),$$

где:

$NP(S)$ - множество именных групп в тексте S.

Для каждой именной группы вычисляется средняя тональность входящих в неё слов (формула 11):

$$sent_{topic} = \frac{1}{|NP|}\sum_{j=1}^{m} sent(NP_j), \quad (11),$$

где:

$sent(NP_j)$ - средняя тональность слов в j-й именной группе.

Также вычисляется тональность всех слов, не входящих в именные группы (формула 12)

$$sent_{context} = \frac{1}{n - |\, NP_{words} \,|} \sum_{i \notin NP} sent(t_i), \quad (12),$$

где:

NP_{words} - множество токенов, входящих в именные группы.

$$J_{sem} = |\, sent_{topic} - sent_{context} \,| \in [0,2], \quad (13),$$

где:

$sent_{topic}$ - тональность темы (о чём текст);

$sent_{context}$ - тональность контекста (что о теме говорят).

Необходимо вычислить коэффициент эмоциоанльной интенсивности (формула 14), так как сарказм часто сопровождается резкими перепадами эмоционального тона [92].

$$\varepsilon_{\text{int}} = \frac{1}{n-1} \sum_{i=1}^{n-1} |\, emo(t_i) - emo(t_{i+1}) \,| * \mathbb{I}[\, emo(t_i) - emo(t_{i+1}) \,| > \tau_e], \quad (14),$$

где:

$emo(t_i) \in [0,1]$ - интенсивность эмоции токена;

$\tau_e \in [0,1]$ - порог чувствительности (обучаемый параметр);

$\mathbb{I}[\cdot]$ - функция принадлежности элемента к подмножеству.

Кроме того необходима оценка преувеличений, характерных для сарказма (формула 15) [93].

$$H_{hyp} = \frac{1}{n} \sum_{i=1}^{n} hyper(t_i) * \mathbb{I}[hyper(t_i) > 0] \in [0,1], \quad (15),$$

где:

$hyper(t_i)$ - вес гиперболичности токена;

n - число токенов.

Стилистический диссонанс - отклонение стиля текста от эталонного для данной темы [94]. Саркастические тексты часто имеют аномальный стиль по сравнению с «нормальными» текстами на ту же тему (формула 16).

$$S_{dis} = \frac{\| H_{syn} - H_{syn}^{base}(k) \|}{\| H_{syn}^{base}(k) \| + \varepsilon} \in [0, \infty), \quad (16),$$

где:

$H_{\text{syn}}^{\text{base}}(k)$ - усредненный синтаксический профиль текстов темы k в обучающей выборке;

$\|\cdot\|$ - евклидова норма;

$\varepsilon = 10^{-8}$ - малая константа, предотвращающая деление на ноль.

Реализует механизм масштабированного скалярного произведения (формула 17) [95].

$$\alpha_i^{sarc} = \text{softmax}(\frac{Q * K_i^T}{\sqrt{d_k}})_i, \quad (17),$$

где:

$Q = W_q \cdot [J_{\text{sem}}, \mathcal{E}_{\text{int}}, D_{\text{parts}}, H_{\text{hyp}}, S_{\text{dis}}]^\top \in \mathbb{R}^{d_k}$ - запрос, агрегирующий все признаки сарказма;

$K_i = W_k \cdot X_{i,:} \in \mathbb{R}^{d_k}$ - ключ для i-го токена;

$V_i = W_v \cdot X_{i,:} \in \mathbb{R}^{d_v}$ - значение для i-го токена;

$W_q \in \mathbb{R}^{d_k \times 5}$, $W_k \in \mathbb{R}^{d_k \times d}$, $W_v \in \mathbb{R}^{d_v \times d}$ - обучаемые матрицы проекций.

Взвешенная сумма эмбеддингов токенов с весами внимания необходима для рассчета вектора контекстной противоречивости (формула 18).

$$CCV = \sum_{i=1}^{n} \alpha_i^{sarc} * V_i \in R^{d_v}, \quad (18),$$

Преобразование вектора контекстной противоречивости в вероятностную оценку наличия сарказма (формула 19):

$$\rho = \sigma(W_\rho * CVV + b_\rho) \in [0,1], \quad (19),$$

где:

$W_\rho \in \mathbb{R}^{1 \times d_v}$, $b_\rho \in \mathbb{R}$ - обучаемые параметры;

σ - сигмоидная функция.

Вычисляется коэффициент контекстного диссонанса (формула 20) [96]:

$$C_{dis} = 1 - \cos(SP_{current}, SP_{context}) \in [0,2], \quad (20),$$

где:

$SP_{current}$ - семантическое пространство текущего сообщения;

$SP_{context} = \frac{1}{m} \sum_{j=1}^{m} SP_j$ - усредненное семантическое пространство предыдущих m сообщений в диалоге;

m - размер окна контекста (гиперпараметр, обычно $m = 3 - 5$).

Семантическое пространство текста представляет собой его интегральную характеристику, получаемую путем анализа взаимосвязей между токенами с учетом их контекстного окружения. Для вычисления семантического пространства применяется метод сингулярного разложения матрицы подстановки токенов. Семантическое пространство текста вычисляется на основе эмбеддингов токенов и отражает его смысловую структуру (формула 21):

$$SP = \sum_{i=1}^{n} a_i \mathrm{X}_i \in R^d \ , \tag{21}$$

где:

SP - взвешенная сумма эмбеддингов;

a_i - степень семантической значимости i-го токена для представления всего текста;

X_i - элемент матрицы эмбеддингов (формула 2).

Коэффициент конвергенции характеризует степень близости семантического пространства анализируемого текста к эталонному пространству, построенному на основе обучающей выборки. Чем выше конвергенция, тем более типичным для данной предметной области является текст. Конвергенция вычисляется как косинусное сходство между векторами семантических пространств и отражает степень сходства с эталоном (формула 22):

$$K = \cos(SP_{new}, SP_{base}) = \frac{SP_{new} * SP_{base}}{| SP_{new} | * | SP_{base} |} , \tag{22}$$

где:

K - коэффициент конвергенции;

SP_{new} - семантическое пространство анализируемого текста;

SP_{base} - базовое семантическое пространство.

Аналогично признакам, семантическое пространство также подвергается нормализации для обеспечения сопоставимости с другими компонентами модели. Нормализация семантического пространства выполняется по той же схеме, что и нормализация признаков (формула 23):

$$SP_{norm} = \frac{SP - \mu_{SP}}{\sigma_{sp} + \in}, \quad (23),$$

где:

μ_{SP} - среднее значение семантического пространства;

σ_{SP} - стандартное отклонение;

ϵ - малая константа для предотвращения деления на ноль.

Интегральный вектор признаков формируется конкатенацией всех извлеченных компонент с регуляризацией аномальностью (формула 24):

$$\Pr iz_{sarc}^{new} = [H_{syn} \oplus H_{sem} \oplus CCV \oplus \rho \oplus J_{sem} \oplus E_{\text{int}} \oplus D_{parts} \oplus H_{hyp} \oplus S_{dis} \oplus C_{dis}] * (1 + \lambda * \delta) \in R^{D_{sarc}}, \quad (24)$$

где:

D_{sarc} - полная размерность;

$\lambda \in \mathbb{R}$ - обучаемый параметр регуляризации аномальностью. При $\lambda = 0$ аномальность игнорируется; при $\lambda > 0$ аномальные тексты усиливают свои признаки;

δ - нормированная аномальность (формула 8);

$\oplus$ - операция конкатенации;

J_{sem} - индекс семантической инконгруэнтности (формула 13);

D_{parts} - косинусное расстояние между частями текста (формула 9).

Для обеспечения численной устойчивости все компоненты вектора $\text{Priz}_{\text{sarc}}^{\text{new}}$ центрируются и масштабируются по статистикам обучающей выборки (формула 25):

$$\Pr iz_{norm} = \frac{\Pr iz_{sarc}^{new} - \mu_{priz}}{\sigma_{\Pr iz} + \varepsilon} \in R^{D_{sarc}}, \quad (25)$$

где:

$\mu_{\text{Priz}} \in \mathbb{R}^{D_{\text{sarc}}}$, $\sigma_{\text{Priz}} \in \mathbb{R}^{D_{\text{sarc}}}$ - среднее и стандартное отклонение по обучающей выборке. Вычисляются на обучающей выборке и сохраняются в артефактах модели. При применении используется сохранённое значение, а не статистика текущего пакета.

Следующий уровень выполняет гибридную категоризацию, объединяя нейросетевую и лингвистическую оценки. Нейросетевой блок применяет

линейный классификатор с softmax-нормализацией к нормализованным признакам, формируя распределение вероятностей по категориям при однометочном режиме (формула 26):

$$neural_{out} = \text{softmax}(W_{cat} * \text{Pr}iz_{norm} + b_{cat}) \in R^{C}, \quad (26)$$

где:

$neural_{out}$ - распределение вероятностей по категориям;

C - количество категорий;

W_{cat} - матрица весов размерности [C×dim(Priz)];

$Priz_{norm}$- вектор признаков;

b_{cat} - вектор смещений размерности [C].

Вводится скалярный коэффициент уверенности на основе конвергенции (формула 27):

$$\kappa = \sigma(K) \in [0,1], \quad (27)$$

где:

σ - сигмоидная функция,

K - коэффициент конвергенции (формула 22).

Результат вычисленний формулы 27 необходимо отнести к нужной категории, для это используется обучаемая матрица, которая позволяет проецировать скалярную оценку в пространство категорий (Формула 28).

$$ling_{vect} = W_{proj} * (SP_{norm} \odot \kappa) \in R^{C}, \quad (28)$$

где:

SP_{norm} - нормализованное семантическое пространство;

$ling_{raw}$ - лингвистическая оценка категории;

$\odot$ - поэлементное (адамарово) умножение;

κ - скаляр, транслируемый на все размерности SP_{norm}.

$W_{proj} \in \mathbb{R}^{C \times d}$- матрица проекции размерности.

Последующее многозадачное обучение описывается формулой 29.

$$combined_{out} = \alpha' * neural_{out} + \beta' * ling_{vect} + \gamma' * (\rho \odot w_{\rho})_{c}, \quad (29)$$

где:

$wp_p \in R^C$ - обучаемый вектор весов влияния саркастичности на каждую категорию;

ρ - скаляр саркастичности, транслируемый на все размерности.

Чтобы объединить нейросетевую и лингвистическую оценки необходимо ввести адаптивные веса. Веса нормируются таким образом, что их сумма равнялась единице (Формула 30)

$$\alpha' = \frac{e^{\alpha}}{e^{\alpha}+e^{\beta}+e^{\gamma}}, \beta' = \frac{e^{\beta}}{e^{\alpha}+e^{\beta}+e^{\gamma}}, \gamma' = \frac{e^{\gamma}}{e^{\alpha}+e^{\beta}+e^{\gamma}} \quad , \tag{30}$$

с условием $\alpha' + \beta' + \gamma' = 1$.

где:

α, β, γ - обучаемые параметры;

α', β', γ' - нормированные веса, но $\alpha' + \beta' + \gamma' = 1$.

Полученная комбинация преобразуется в распределение вероятностей с помощью функции «softmax» при одноточном режиме (Формула 31):

$$cat_{probs} = \text{softmax}(combined_{out}) \in R^C, \tag{31}$$

Итоговая категория определяется как класс с максимальной вероятностью (Формула 32) при одноточном режиме, при многоточном используется сигмоида (Формула 32.1):

$$cat = \arg\max_{C \in \{1,\dots C\}} (cat_{probs}), \tag{32}$$

$$cat = \{c : \sigma(combined_out)_c > \tau_c\}, \tag{32.1}$$

Сет S допустим для множества категорий Cat, если он покрывает все предсказанные категории и не противоречит аксиомам и правилам онтологии (Формула 33)

$$Adm(S, Cat, \Omega) = 1[Cat \subseteq S.cat] * 1[\forall \alpha \in A: \alpha(S, Cat) = true] * 1[\forall \theta \in \Theta: \theta(S, Cat) = true], \tag{33}$$

где:

$\mathbf{1}[\cdot]$ - индикаторная функция (1 - истина, 0 - ложь),

S - сет (набор категорий/понятий онтологии),

Cat - множество предсказанных категорий,

$\Omega = (A, \Theta)$ - онтология, где A - множество аксиом, Θ - множество правил,

$S.cat$- множество категорий, покрываемых сетом,

$a(S, Cat)$ - проверка аксиомы для сета и категорий,

$\theta(S, Cat)$ - проверка правила для сета и категорий.

Среди всех допустимых сетов выбирается максимальный, который покрывает наибольшее число предсказанных категорий (Формула 34).

$$Set = arg\max_{S \in Sets} \sum_{c \in Cat} 1[c \in S.cat] * Adm(S, Cat, \Omega), \quad (34)$$

где:

$Sets$ - множество всех рассматриваемых сетов (наборов категорий онтологии),

Cat - множество предсказанных категорий,

$S.cat$- множество категорий, покрываемых сетом,

$\mathbf{1}[\cdot]$ - индикаторная функция (1 - истина, 0 - ложь),

$\Omega = (A, \Theta)$ - онтология, где A - множество аксиом, Θ - множество правил,

$Adm(S, Cat, \Omega)$ - предикат допустимости сета.

Если несколько сетов имеют одинаковую мощность, выбирается тот, у которого выше суммарная вероятность покрытых категорий (Формула 34.1)

$$Set = arg\max_{S \in Sets} \sum_{c \in Cat \cap S.cat} cat_prob[c] * Adm(S, Cat, \Omega), \quad (34.1)$$

где:

$Sets$ - множество всех рассматриваемых сетов (наборов категорий онтологии),

Cat - множество предсказанных категорий,

$S.cat$- множество категорий, покрываемых сетом,

$c \in Cat \cap S.cat$ - категории, которые и предсказаны, и покрыты сетом,

$cat_prob[c]$ - вероятность категории,

$\Omega = (A, \Theta)$ - онтология, где A - множество аксиом, Θ - множество правил,

$Adm(S, Cat, \Omega)$ - предикат допустимости сета.

Уверенность сета определяется как минимум вероятностей категорий, вошедших в сет (Формула 35).

$$conf(set) = \min_{c \in Cat \cap S.cat} cat_prob[c], \quad (35)$$

где:

Set - выбранный сет,

Cat - множество предсказанных категорий,

$S.cat$- множество категорий, покрываемых сетом,

$c \in Cat \cap S.cat$ - категории, которые и предсказаны**,** и покрыты сетом,

$cat_prob[c]$ - вероятность категории,

$conf(Set)$ - уверенность сета (минимальная вероятность среди покрытых предсказанных категорий)

Если ни один сет не является допустимым формируется сет «Прочее» (Формула 36).

$$\forall S \in Sets: Adm(S, Cat, \Omega)=0\ , \quad (36)$$

где:

Set - выбранный сет,

Cat - множество предсказанных категорий,

$\Omega = (A, \Theta)$ - онтология, где A - множество аксиом, Θ - множество правил,

$Adm(S, Cat, \Omega)$ - предикат допустимости сета.

Итоговая бинарная классификация сарказма (Формула 37, 38):

$$\mathrm{SPr} = \mathrm{softmax}(W_{sarc} * \mathrm{Pr}iz_{norm} + b_{sarc}) \in R^2\ , \quad (37)$$

$$SCM = \arg\max_{k \in \{0,1\}} (\mathrm{SPr})_k\ , \quad (38)$$

где:

$W_{\mathrm{sarc}} \in \mathbb{R}^{2 \times D_{\mathrm{sarc}}}$, $b_{\mathrm{sarc}} \in \mathbb{R}^2$ - обучаемые параметры,

$k = 0$ - сарказм отсутствует,

$k = 1$ - сарказм присутствует.

Контент получает оценку степени тяжести от 0 до 4 независимо от категории. Вектор для этой оценки строится из сочетания признаков (Формула 39):

$$sev = \mathrm{Priz}_{clean} \oplus H_{ton} \oplus \delta \oplus \sigma(K) \oplus SP_{norm} \oplus \log(1 + \Delta_B) \oplus J_{sem} \oplus \varepsilon_{\mathrm{int}} \oplus \rho \in R^{D_{sev}}\ , \quad (39)$$

где:

$D_{\mathrm{sev}} = (h_{\mathrm{syn}} + h_{\mathrm{sem}}) + 3 + 1 + 1 + d + 1 + 1 + 1 + 1$ - полная размерность;

$\log(1+\Delta_B)$ - сглаживание для подавления выбросов при больших частотах;

$\mathcal{I}_{\text{sem}}$ - индекс семантической инконгруэнтности;

ρ - скаляр саркастичности.

Непрерывная оценка тяжести (Формула 40):

$$s = W_{sev} * sev_input + b_{sev} \in R, \tag{40}$$

где:

$W_{\text{sev}} \in \mathbb{R}^{1\times D_{\text{sev}}}$, $b_{\text{sev}} \in \mathbb{R}$ - обучаемые параметры.

Порядковая регрессия с обучаемыми порогами переводит непрерывную оценку в шкалу 0-4 (Формула 41):

$$\theta_1 < \theta_2 < \theta_3 < \theta_4, \theta_k \in R, \tag{41}$$

Чтобы получить логиты для каждого уровня, вычитаем непрерывную оценку из порогов (Формула 42):

$$l_k = -(s-\theta_k)^2, k = 0,1,2,3,4 \tag{42}$$

где:

$\theta_0 = -\infty$ (практически используется $\theta_0 = -10^3$),

θ_4 - верхний порог.

Полный вектор логитов имеет вид, представленный формулой 43:

$$L = \left[l_0, l_1, l_2, l_3, l_4\right]^T \in R^5, \tag{43}$$

Логиты нормализуются формулой 44 получается вектор вероятностей. Значение итоговой степени принимает индекс максимального элемента этого вектора (Формула 45).

$$sev_{probs} = \text{softmax}(L) \in R^5, \tag{44}$$

$$sev = \arg\max_{k\in\{0,1,2,3,4\}} (sev_{probs})_k, \tag{45}$$

Для коротких сообщений (до n_0 токенов) увеличивается вес локальных признаков и уменьшается вес глобальных (формула 46):

$$\text{Priz}_{short} = [\alpha(n) * \text{Priz}_L \,||\, \beta(n) * \text{Priz}_G], \tag{46}$$

где:

$Priz_L$ - локальные признаки;

$Priz_G$ - глобальные признаки;

$\alpha(n) = \sigma\left(\frac{n-n_0}{\tau}\right)$ - функция адаптации локальных признаков;

$\beta(n) = 1 - \alpha(n)$ - функция адаптации глобальных признаков;

n - длина сообщения (число токенов);

$n_0 = 20$ - порог короткого сообщения;

$\tau = 10$ - гиперпараметр;

σ - сигмоидная функция.

Разработанный алгоритм (Приложение Б) квалификации неструктурированных данных представляет собой формальную последовательность действий, реализующую функцию квалификации текстов с детекцией сарказма, определенную в формальной модели (раздел 2.1). Входными данными алгоритма является исходный текст, выходными - категория текста, бинарный флаг наличия сарказма, оценка степени тяжести от нуля до четырех, а также вероятностные распределения по всем выходным параметрам. Алгоритм включает десять последовательных этапов, каждый из которых имеет четко определенные входные данные, условия выполнения и выходные результаты.

Первый этап - токенизация и векторное представление. Прмиеняется алгоритм Byte Pair Encoding и выполняется процедуа разбиения теста на посдлова, в результате чего текст представляется в виде последовательности целочисленных идентификаторов токенов из словаря. Далее осуществляется построение эмбеддингов токенов посредством матрицы подстановки, что позволяет перейти от дискретных символов к непрерывному числовому пространству. Результатом этапа являются последовательность токенов и матрица эмбеддингов. Условием перехода к следующему этапу является успешное формирование матрицы эмбеддингов.

Второй этап - извлечение базовых признаков текста. На данном этапе из матрицы эмбеддингов и токенизированного текста извлекаются четыре группы характеристик, вычисляемых параллельно. Синтаксические признаки отражают структурно-грамматические особенности текста и извлекаются с использованием одномерной сверточной нейронной сети с фильтрами различных размеров. Семантические признаки характеризуют смысловое содержание и тематическую

принадлежность текста и извлекаются методом латентного размещения Дирихле. Тональная характеристика определяется посредством гибридного подхода, сочетающего лексический анализ на основе словаря тональных маркеров и нейросетевой классификатор. Показатель аномальности вычисляется на основе дельты Берроуза с последующим нормированием в диапазон от нуля до единицы и служит для выявления потенциальных информационных вбросов. Результатом этапа являются четыре группы признаков. Условием перехода к следующему этапу является успешное извлечение всех четырех групп признаков.

Третий этап - вычисление улучшенных лингвистических признаков. Извлекаются дополнительные лингвистические признаки, специфические для детекции сарказма. Вычисляется косинусное расстояние между частями текста для количественной оценки внутреннего конфликта между началом и концом предложения. Определяется индекс семантической инконгруэнтности как абсолютная разность между тональностью темы, представленной именными группами, и тональностью контекста. Вычисляется коэффициент эмоциональной интенсивности, отражающий контраст между эмоционально заряженными фрагментами текста. Определяется индекс гиперболизации, оценивающий использование преувеличений. Вычисляется коэффициент стилистического диссонанса, сравнивающий стиль текста с эталонным стилем для данной темы. Результатом этапа являются пять признаков. Условием перехода к следующему этапу является успешное вычисление всех пяти признаков.

Четвертый этап - механизм внимания для выделения индикаторов сарказма. Реализуется механизм внимания, основанный на комбинации всех ранее вычисленных признаков. Формируется запрос, агрегирующий все признаки сарказма, для каждого токена вычисляются ключ и значение на основе его эмбеддинга через обучаемые матрицы проекций. Вычисляются веса внимания для каждого токена, отражающие важность токена для определения сарказма. На основе весов формируется вектор контекстной противоречивости как взвешенная сумма значений токенов, который затем преобразуется в скалярную вероятностную оценку наличия сарказма. Результатом этапа являются массив весов внимания,

вектор контекстной противоречивости и скаляр саркастичности. Условием перехода к следующему этапу является успешное вычисление всех трех компонентов.

Пятый этап - вычисление контекстного диссонанса для диалогов. Вычисляется коэффициент контекстного диссонанса как косинусное расстояние между семантическим пространством текущего сообщения и усредненным семантическим пространством предыдущих сообщений в диалоге. Размер окна контекста составляет от трех до пяти предыдущих сообщений. В случае отсутствия контекста данный этап может быть пропущен с сохранением нейтрального значения коэффициента. Результатом этапа является коэффициент контекстного диссонанса. Условием перехода к следующему этапу является успешное вычисление коэффициента при наличии контекста или пропуск этапа при его отсутствии.

Шестой этап - вычисление семантического пространства и конвергенции. На данном этапе выполняется построение семантического пространства текста как взвешенной суммы эмбеддингов токенов. Вычисляется коэффициент конвергенции как косинусное сходство между семантическим пространством анализируемого текста и базовым семантическим пространством, построенным как центроид семантических пространств обучающей выборки. Выполняется нормализация семантического пространства путем вычитания среднего и деления на стандартное отклонение. Результатом этапа являются семантическое пространство, коэффициент конвергенции и нормализованное семантическое пространство. Условием перехода к следующему этапу является успешное формирование нормализованного семантического пространства.

Седьмой этап - формирование интегрального вектора признаков и нормализация. В рамках этапа выполняется конкатенация всех извлеченных компонентов с последующим умножением на регуляризационный множитель, зависящий от показателя аномальности, что позволяет усиливать признаки аномальных текстов и ослаблять признаки типичных текстов. Выполняется Z-стандартизация интегрального вектора путем вычитания среднего и деления на

стандартное отклонение. Результатом этапа являются расширенный интегральный вектор и нормализованный интегральный вектор. Условием перехода к следующему этапу является успешная нормализация интегрального вектора.

Восьмой этап - адаптация для коротких сообщений. На этом этапе выполняется синтез признаков для коротких текстов путем умножения расширенного интегрального вектора на адаптационный коэффициент. Коэффициент вычисляется с использованием сигмоидной функции от разности длины сообщения и порогового значения. При длине сообщения значительно меньше порогового значения коэффициент стремится к нулю, что уменьшает влияние глобальных признаков и усиливает роль локальных характеристик. При длине сообщения значительно больше порогового значения коэффициент стремится к единице. Результатом этапа являются адаптированные признаки, используемые для повышения точности классификации коротких сообщений. Конечное условие выполнения алгоритма - успешное формирование всех выходных компонентов: категории, бинарного флага сарказма, оценки степени тяжести и соответствующих вероятностных распределений.

Девятый этап - гибридная категоризация при которой запускаются две параллельные ветви обработки. В нейросетевой ветви линейный классификатор с softmax-нормализацией применяется к нормализованному интегральному вектору. В лингвистической ветви на основе коэффициента конвергенции вычисляется лингвистический коэффициент уверенности, затем нормализованное семантическое пространство масштабируется на данный коэффициент и проецируется в пространство категорий. Для объединения ветвей вычисляются адаптивные веса гибридизации через softmax-нормализацию трех обучаемых логитов. Гибридная оценка определяется как взвешенная сумма нейросетевой оценки, лингвистической векторной оценки и скаляра саркастичности. Полученная комбинация преобразуется в распределение вероятностей по категориям, итоговая категория определяется как класс с максимальной вероятностью. Параллельно выполняется бинарная классификация сарказма путем применения линейного классификатора к нормализованному интегральному вектору. Результатом этапа

являются распределение вероятностей по категориям, итоговая категория, распределение вероятностей сарказма и бинарный флаг наличия сарказма. Условием перехода к следующему этапу является успешное формирование всех выходных компонентов.

Десятый этап - оценка степени тяжести. На данном этапе выполняется формирование входного вектора для оценки тяжести путем конкатенации очищенного интегрального вектора, вектора тональности, показателя аномальности, сигмоиды от коэффициента конвергенции, нормализованного семантического пространства, логарифмически сглаженной дельты Берроуза, индекса семантической инконгруэнтности, коэффициента эмоциональной интенсивности и скаляра саркастичности. Линейный слой преобразует полученный вектор в непрерывную скалярную оценку. Для перевода непрерывной оценки в дискретные уровни от нуля до четырех используется система обучаемых порогов, упорядоченных по возрастанию. Для каждого уровня вычисляется квадратичный логит как отрицательный квадрат разности между непрерывной оценкой и соответствующим порогом. Вектор логитов преобразуется в распределение вероятностей по уровням тяжести, итоговый уровень определяется как индекс максимального элемента. Результатом этапа являются распределение вероятностей по уровням тяжести и итоговый уровень тяжести. Условием перехода к следующему этапу является успешное формирование оценки тяжести.

Алгоритм охватывает полный цикл квалификации - от токенизации исходного сообщения до отнесения его к категории с одновременной детекцией сарказма и оценкой степени тяжести. Последовательность из десяти шагов построена так, что каждый предыдущий этап подготавливает данные для последующего. Выходные компоненты (категория, флаг сарказма, степень тяжести, вероятностные распределения) формируются только после успешного завершения всех операций.

Математическое представление алгоритма квалификации отличается комплексным объединением контекстуализированных векторных представлений, тонального анализа и модифицированной дельты Берроуза на уровне признакового

пространства, а также использованием порядковой регрессии с обучаемыми порогами для раздельной оценки степени тяжести. Алгоритм включает десять последовательных этапов - от токенизации и извлечения базовых признаков до гибридной категоризации и оценки тяжести.

3.4 Выводы по главе 3

Проведённый в третьей главе анализ методов парсинга и обработки данных из социальных медиа, разработка базовых алгоритмов семантического и тонального анализа, а также формализация алгоритма квалификации неструктурированных данных позволили определить математический аппарат и инструментальные средства, обеспечивающие полный цикл квалификации текстового контента.

1. Семантический анализ как этап обработки естественного языка выполняет информационную, описательную, структурную и преобразовательную функции, а его процесс включает сбор данных, предварительную обработку, определение семантических ролей, выявление связей и обработку результатов. Эффективность семантического анализа зависит от качества предобработки и выбора подхода к семантическим ролям, а разделение контента на целевой и шумовой требует дополнительных критериев, что определяет необходимость формализации базовых алгоритмов обработки текста.

2. Алгоритм BPE-токенизации обеспечивает эффективное разбиение текста на подсловные единицы, решает проблему редких и неизвестных слов, а byte-level модели дополнительно обеспечивают кросс-лингвистичность и устойчивость к лингвистическому шуму. Однако BPE не учитывает семантику и контекст, поскольку токены формируются по частотному критерию, что ограничивает его применение в задачах, требующих понимания смысловых связей, и обосновывает необходимость дополнения BPE алгоритмами семантического и тонального анализа. Рассмотренные алгоритмы семантического и тонального анализа обеспечивают извлечение смысловых отношений, построение графа знаний и определение эмоциональной окраски текста, однако каждый из них в отдельности

имеет ограничения: семантический анализ требует ресурсоёмкой предобработки и зависит от качества грамматических парсеров, а тональный анализ чувствителен к сарказму, иронии и контекстуальной многозначности.

3. Разработано математическое представление алгоритма квалификации, отличающееся комплексным объединением контекстуализированных векторных представлений, тонального анализа и модифицированной дельты Берроуза на уровне признакового пространства, а также использованием порядковой регрессии с обучаемыми порогами для раздельной оценки степени тяжести. Алоритм программного решения

Таким образом сочетание трёхуровневой иерархии и вложенных графов обеспечивает одновременное сохранение интерпретируемости на верхнем уровне и гибкости на нижних; байтовая токенизация обеспечивает кросс-лингвистичность и устойчивость к шуму; дельта Берроуза служит инструментом выявления аномалий; порядковая регрессия с обучаемыми порогами - инструментом раздельной оценки степени тяжести. Совокупность этих решений образует математический аппарат и инструментальную основу для программной реализации, рассматриваемой в главе 4.

4 СТРУКТУРА ПРОГРАММНОГО РЕШЕНИЯ КВАЛИФИКАЦИИ И ЕЕ РЕАЛИЗАЦИЯ В ПАРСИНГОВОЙ СИСТЕМЕ

4.1 Анализ объекта внедрения и существующий механизм квалификации контента

В рамках настоящего исследования парсинг рассматривается как комплексный процесс, включающий синтаксический анализ и автоматизированное извлечение данных из общедоступных информационных ресурсов. Парсинговые системы, также известные как парсеры, представляют собой специализированные программные комплексы, осуществляющие автоматизированную обработку открытых источников информации. Под открытыми информационными ресурсами понимаются медиаплатформы с общедоступным контентом на его использование,

включая средства массовой информации, научные публикации, веб-ресурсы и социальные медиа.

На международном рынке представлены парсинговые решения для аналитических задач, среди которых - «Brandwatch» [114] и «Brand Analytics» [115]. Интеграция таких методов в системы анализа бизнес-процессов повышает детализацию данных, уточняет оценку рыночных тенденций и улучшает прогнозирование. Общий принцип работы парсинговых систем показан на рисунке 12.

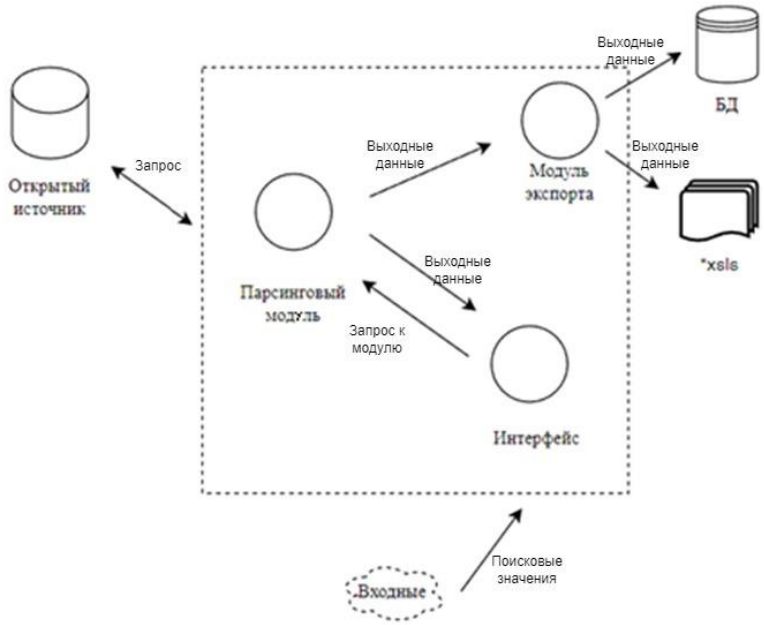


Рисунок 12 - Концептуальная схема работы парсинговых систем

Пользователь инициирует запрос через интерфейс системы. Запрос преобразуется в кодированную форму и направляется к целевому открытому источнику. Полученный ответ обрабатывается и визуализируется для пользователя; параллельно данные сохраняются в централизованной базе данных.

Результаты экспортируются в структурированных таблицах, текстовых документах, размеченном контенте или комбинированном формате - такая гибкость адаптирует данные под разные аналитические задачи.

Архитектура внедрения реализована на базе SaaS-платформы, интегрирующей квалификационный метод обработки. Инфраструктура -

распределённый серверный кластер, оптимизированный для производительности, отказоустойчивости и информационной безопасности.

Вычислительная инфраструктура включает сеть из десяти прокси-серверов для непрерывного мониторинга интернет-пространства. Их работу координирует сервер управления, планирующий задачи и отслеживающий состояние компонентов. Данные поступают на сервер промежуточного хранения, который распределяет их между серверами баз данных. Последние выполняют индексацию и формируют единую базу через автоматическое сопоставление.

Первичная обработка осуществляется специализированным сервером, который применяет заданные алгоритмы анализа и передает отобранные данные в нейросетевой комплекс. Серверы искусственного интеллекта выполняют углубленную обработку, результаты которой сохраняются в базе данных выдачи. При этом система динамически корректирует приоритеты мониторинга через взаимодействие с сервером управления.

Аналитический модуль интегрирован с сервером искусственного интеллекта и обеспечивает двусторонний обмен данными. С одной стороны, он получает обработанную информацию для работы аналитиков и передачи в соответствующие инстанции; с другой - передает результаты аналитической деятельности для дальнейшего обучения нейронной сети. Конкурентное преимущество системы в которую внедряется решение - комплексный подход к обработке данных, включающий не только мониторинг и статистический анализ, но и углубленную аналитику контента.

Модель текущего процесса квалифкации данных, представленна на рисунке 13. Через интерфейс управления пользователь задает параметры поискового запроса: семантические маркеры, перечень источников, временной интервал и критерии фильтрации.

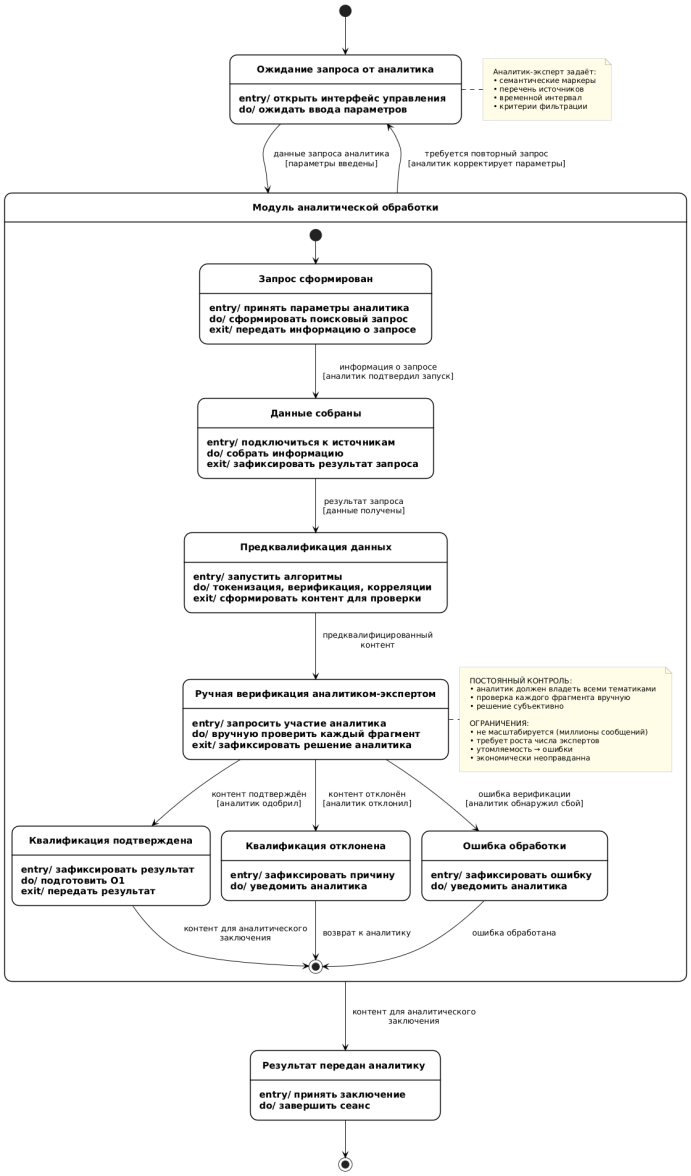


Рисунок 13 - Модель текущего состояния процесса квалифкации

После формирования запроса система обращается к открытым источникам. Полученные данные не структурированы и не классифицированы; их дальнейшая обработка включает комплекс алгоритмических процедур.

Предварительная квалификация состоит из токенизации, словарной верификации и программно-определяемых корреляций. В текущей реализации контекст учитывается ограниченно, а модуль тонального анализа отсутствует - эти ограничения определяют направления дальнейшего развития.

Существующий механизм квалификации требует постоянного контроля со стороны аналитика-эксперта, который должен владеть всеми тематиками, обрабатываемыми системой. При работе с большими данными (миллионы

сообщений) ручная верификация становится неэффективной: она требует значительных временных затрат, не позволяет обрабатывать данные в темпе их поступления и экономически неоправданна - увеличение объёма данных требует пропорционального роста числа экспертов. Кроме того, разные специалисты могут по-разному интерпретировать один и тот же контент, что приводит к несогласованности результатов, а неизбежные ошибки (утомляемость, невнимательность, недостаточная компетентность в отдельных тематических областях) снижают надёжность квалификации.

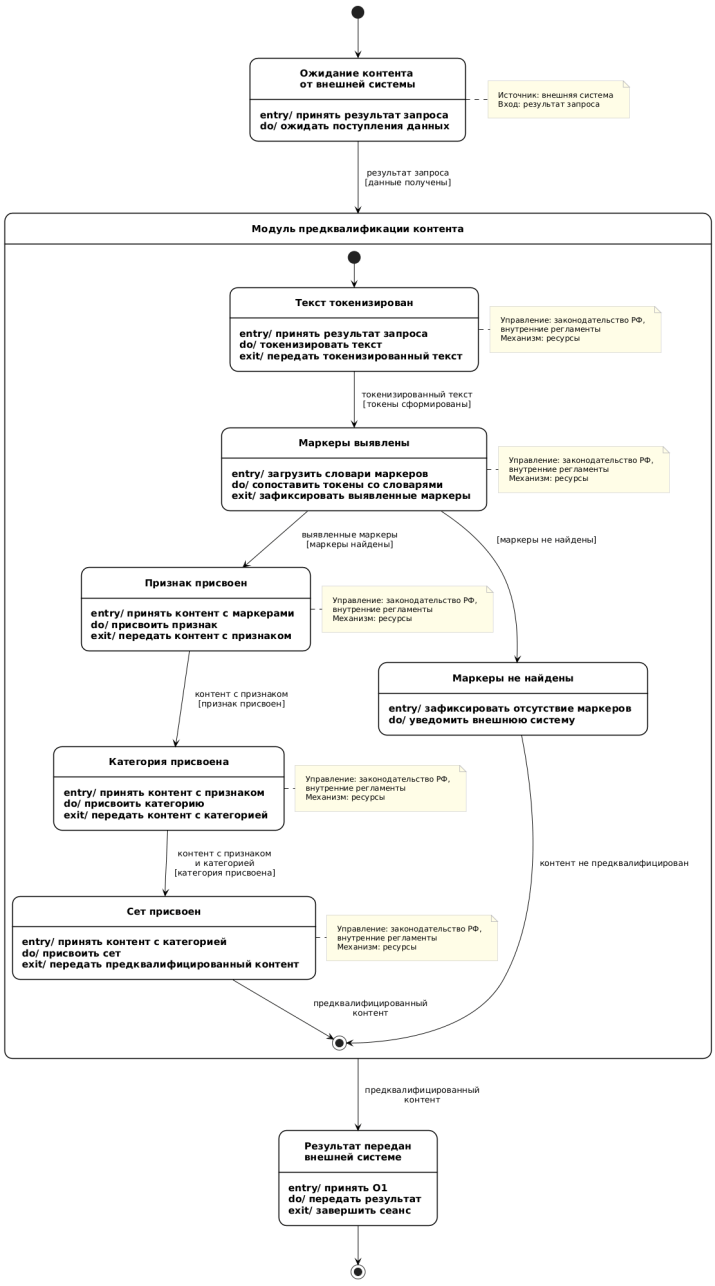


Рисунок 14 - Предквалификация данных»

Представленная модель (Рисунок 14) демонстрирует базовый функционал квалификации контента парсинговой системы, при этом выявляются потенциальные возможности для расширения аналитических возможностей посредством интеграции дополнительных модулей обработки и классификации информации.

Анализ парсинговой системы определяет следующие направления доработки. Внедрение гибридной модели квалифкации данных сократит ручной труд, оставив экспертам контроль неоднозначных случаев и дообучение при появлении новых категорий. Тональный анализ, отсутствующий в текущей реализации, дополнит оценку контента эмоциональной составляющей. Учет контекста вместо чисто словарной верификации - интеграция семантического и синтаксического разбора - обеспечит корректную обработку многозначных выражений и синонимии. Объяснение решений на основе онтологии сделает выводы системы интерпретируемыми для эксперта. Наконец, дельта Берроуза как мера семантического сдвига позволит обнаруживать дрейф языка и предметной области, запуская дообучение моделей и актуализацию онтологии без полной реконструкции.

Реализация этих направлений позволит преобразовать парсинговую систему из инструмента сбора данных в аналитическую платформу, которая не только извлекает, но и интерпретирует содержание пользовательского контента, предоставляя аналитикам понятные и достоверные результаты квалификации.

Регламентация процессов парсинговой деятельности осуществляется в рамках действующего законодательства. Нормативно-правовая база является системой управляющих воздействий, ограничивающих и направляющих функционирование парсинговых систем.

Основополагающим нормативным актом в сфере обработки персональных данных является Федеральный закон от 27.07.2006 № 152-ФЗ «О персональных данных» [121], который устанавливает принципы и условия обработки персональных данных граждан Российской Федерации. В соответствии с данным законом, обработка персональных данных, включая их сбор посредством

парсинговых систем, допускается только при наличии согласия субъекта персональных данных, за исключением случаев, прямо предусмотренных законодательством. Закон также определяет права субъектов персональных данных, включая право на доступ к своим данным, право на их исправление, право на удаление (право на забвение) и право на обжалование действий операторов.

Федеральный закон от 27.07.2006 № 149-ФЗ «Об информации, информационных технологиях и о защите информации» [122] определяет базовые принципы регулирования информационных отношений в сети Интернет, включая обязанности владельцев информационных ресурсов по мониторингу и ограничению доступа к противоправному контенту, а также порядок взаимодействия с органами государственной власти. Данный закон устанавливает правовой режим информации, разграничивая общедоступную информацию и информацию ограниченного доступа.

Федеральный закон от 07.07.2003 № 126-ФЗ «О связи» [123] закрепляет правовую основу деятельности провайдеров услуг связи, включая операторов социальных сетей, и предусматривает возможность ограничения доступа к информационным ресурсам в целях обеспечения общественной безопасности.

В контексте защиты несовершеннолетних от негативного информационного воздействия важное значение имеет Федеральный закон от 29.12.2010 № 436-ФЗ «О защите детей от информации, причиняющей вред их здоровью и развитию» [124], устанавливающий категориальные ограничения на распространение определенного вида контента, способного нанести ущерб нравственному и физическому развитию несовершеннолетних.

Федеральный закон от 01.05.2019 № 90-ФЗ «О внесении изменений в Федеральный закон «О связи» [125] направлен на обеспечение устойчивого функционирования российского сегмента сети Интернет и устанавливает дополнительные требования к операторам связи и владельцам информационных ресурсов.

В контексте автоматизированного сбора данных из социальных медиа особое значение приобретают пользовательские соглашения и политики

конфиденциальности соответствующих платформ. Согласно которой размещенная пользователями информация становится доступной иным пользователям сети Интернет.

Законодательные ограничения (152-ФЗ, 149-ФЗ) налагают требования на автоматические системы обработки контента: владельцы ресурсов обязаны фильтровать противоправные материалы, защищать персональные данные, соблюдать возрастные ограничения и не допускать распространения информации, нарушающей права граждан. Нарушение этих требований влечёт юридическую ответственность. Административная ответственность регламентируется Кодексом об административных правонарушениях и предусматривает штрафы для граждан, должностных лиц, индивидуальных предпринимателей и юридических лиц - их размер зависит от характера нарушения и последствий. Уголовная ответственность наступает при наличии квалифицирующих признаков, предусмотренных Уголовным кодексом. Таким образом, для систем автоматической квалификации контента юридические риски являются не периферийным, а системообразующим фактором, определяющим требования к точности, интерпретируемости и документированию решений.

Надзор за соблюдением законодательства в сфере обработки информации осуществляет Роскомнадзор. Проверяется законность сбора данных, регистрация операторов персональных данных, соответствие обработки установленным требованиям, рассматриваются жалобы пользователей. Владельцы социальных сетей обязаны идентифицировать пользователей при достижении порогового значения аудитории. Пользователь вправе обжаловать блокировку контента: сначала - владельцу социальной сети, затем, при несогласии в Роскомнадзор или суд. Судебная защита включает возмещение убытков, компенсацию морального вреда, защиту чести, достоинства и деловой репутации.

В международной практике регулирования социальных медиа наблюдаются схожие тенденции. В США действует Закон об исключении ответственности за содержание [126], в Европейском союзе - Регламент об общей защите данных

(GDPR) [127], в Китае - Закон о кибербезопасности [128], а в Германии - Закон о сетевой информации (NetzDG) [129].

В современных условиях наблюдается тенденция к усилению государственного контроля над информационными потоками в социальных медиа, что обусловлено необходимостью обеспечения информационной безопасности, защиты прав граждан и противодействия распространению противоправного контента.

Парсинг представляет собой комплексный процесс извлечения данных из открытых источников, а существующие решения подтверждают востребованность таких систем. Рассмотрены концептуальная схема, архитектура внедрения на базе SaaS-платформы, инфраструктура и модель текущего процесса квалифкации. Установлено, что существующий механизм квалификации основан на словарном подходе и характеризуется отсутствием контекстного и тонального анализа, высокой зависимостью от ручного контроля, низкой точностью при полисемии и ограниченной масштабируемостью. Анализ нормативно-правовой базы подтвердил соответствие разрабатываемого решения требованиям № 149-ФЗ и № 152-ФЗ.

4.2 Реализация программного решения и сценариев развертывания

4.2.1 Реализация подключаемого программного модуля

Архитектура программного решения это упрощённое представление, в котором определяют основные части программы и как они будут взаимодействовать друг с другом и внешним миром.

Диаграмма компнентов UML предназначенна для визуализации организации и зависимостей между компонентами программной системы. По сути, это схема высокого уровня абстракции, отображающая структурные взаимосвязи между частями программного продукта. Основная функция диаграммы компонентов - показать, как программные элементы организованы в модули и подсистемы, а также как эти модули взаимодействуют между собой через интерфейсы.

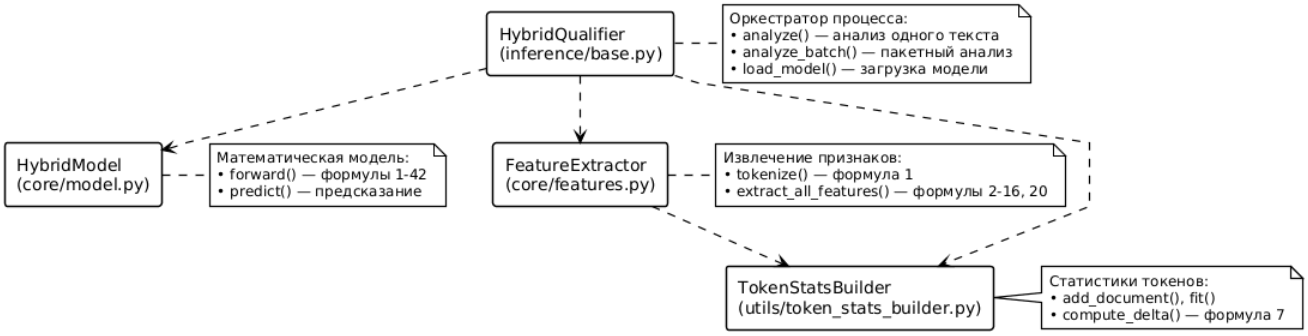


Рисунок 15 – Диаграммы компонентов библиотеки

Диаграмма пакетов (Рисунок 15) отражает модульную структуру библиотеки «hybrid_qualifier», состоящей из пяти функциональных пакетов («api», «core», «inference», «models», «utils») и двух корневых модулей («init.py», «version.py»). Пакет «core» содержит нейросетевую модель и экстрактор признаков, реализующие ядро гибридного анализа. Пакет «inference» предоставляет класс-фасад HybridQualifier, который объединяет модель и экстрактор в единый пайплайн инференса, а пакет models отвечает за загрузку и сохранение предобученных весов. Пакет «api» реализует HTTP-клиент HybridQualifierClient и схемы данных для удалённого взаимодействия, тогда как utils содержит вспомогательные механизмы - пользовательские исключения и построитель статистик токенов TokenStatsBuilder. Зависимости между пакетами направлены строго сверху вниз (от «api» и «inference» к «core» и «utils»), что обеспечивает слабую связность, изоляцию домена от инфраструктуры и возможность повторного использования ядра без изменений.

Фрагмент диаграммы классов (Рисунок 16) представляет статическую структуру библиотеки hybrid_qualifier. Она предоставляет визуальное представление классов, их атрибутов, методов и взаимосвязей между ними. Полное описание классов «hybrid model», «hybrid qualifier», «feature extractor», «token stats builder» представлено в приложении В. Диаграммы классов являются фундаментальным инструментом для проектирования и общения в системах программного обеспечения, что делает их необходимым навыком для разработчиков и архитекторов систем.

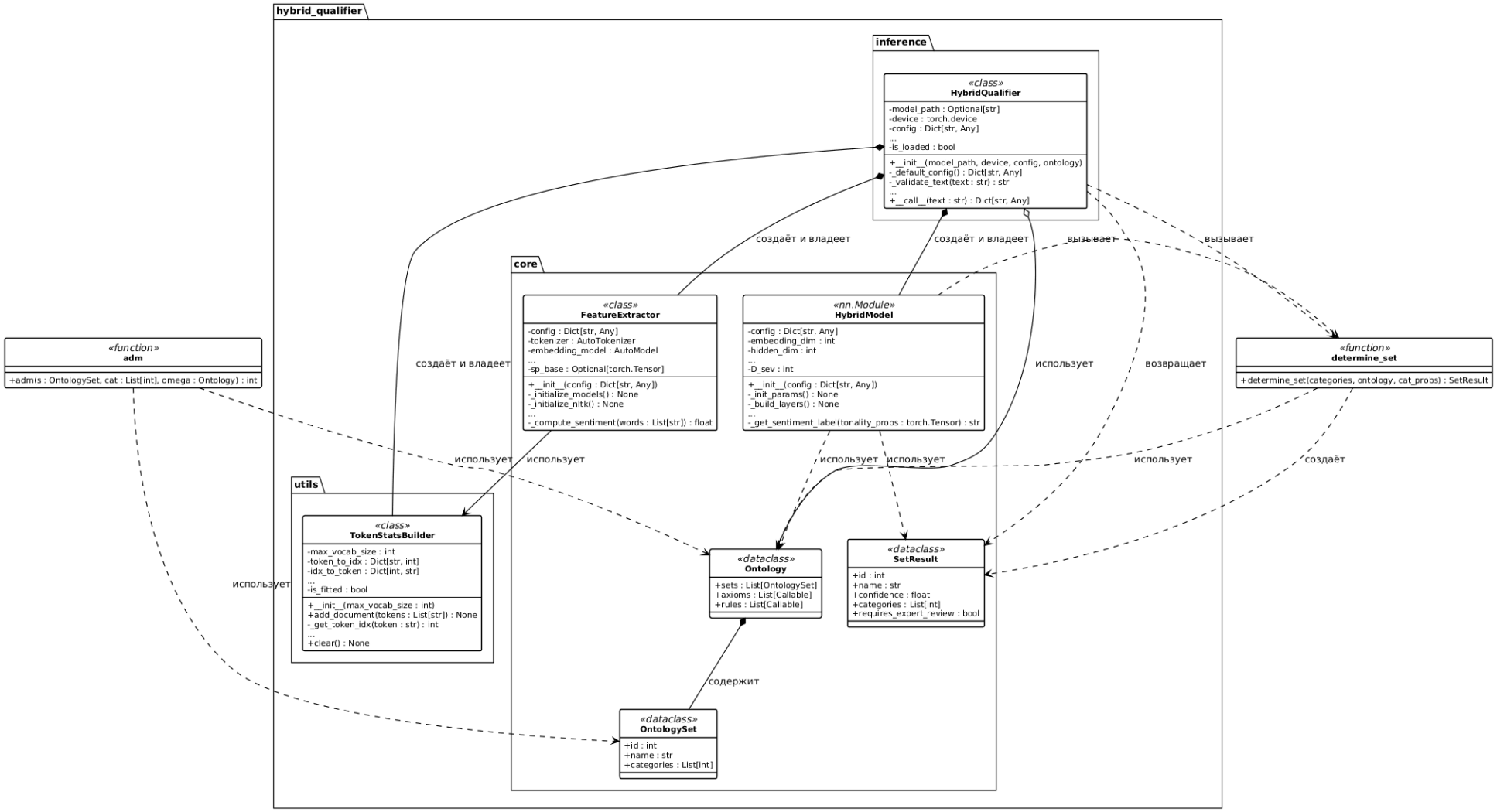


Рисунок 16 - Диаграмма классов библиотеки (фрагмент)

Диаграмма развертывания иллюстрирует физическое развертывание программных компонентов на аппаратных узлах. Она предоставляет обзор архитектуры системы, показывая, как программные модули, базы данных и другие элементы распределены между серверами, компьютерами и устройствами.

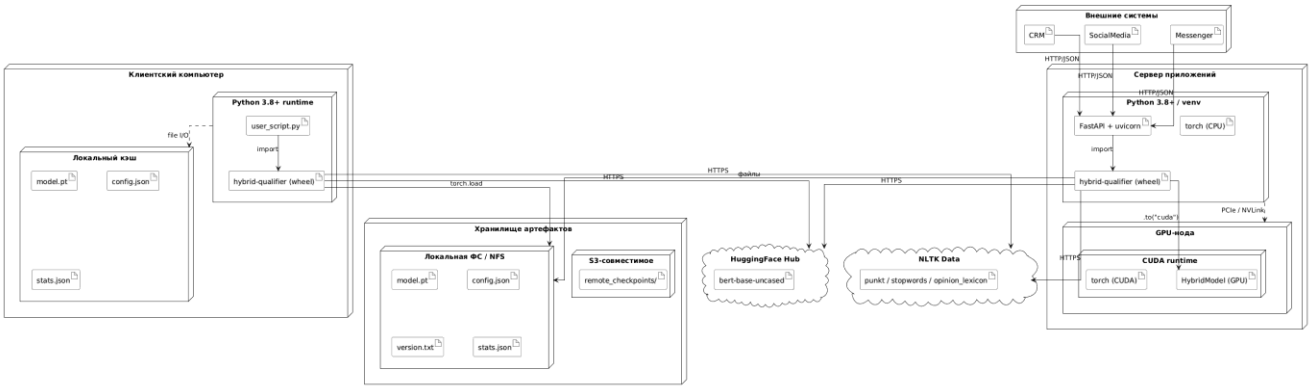


Рисунок 17 - Диаграмма развертывания библиотеки

Диаграмма развёртывания (Рисунок 17) отражает физическое размещение компонентов гибридной системы квалификации текста. Взаимодействие между узлами реализовано по следующему принципу: парсер передаёт тексты в механизм взаимодействия через фасад протокола; механизм взаимодействия вызывает библиотеку квалификации в том же процессе через адаптер; библиотека обращается к хранилищу артефактов (веса модели, конфигурация, файл версии, статистики токенов) посредством файловых операций, к внешнему репозиторию моделей - по HTTPS для загрузки предобученного токенизатора и весов, к внешнему хранилищу лингвистических ресурсов - по HTTPS для словарей и корпусов; результаты квалификации сохраняются в базу данных; аналитический модуль читает их из БД и через REST API предоставляет внешним потребителям. Такая схема обеспечивает минимальные накладные расходы на интеграцию (за счёт встраивания библиотеки в тот же процесс, что и механизм взаимодействия) и возможность горизонтального масштабирования за счёт дублирования узлов парсинга и квалификации без изменения базовой архитектуры сбора данных.

Подключаемое программное решение реализован в виде библиотеки Python (Приложение Г). Библиотека предоставляет функции для загрузки и предобработки текстов, контекстного и тонального анализа, построения семантических графов и итоговой классификации на основе гибридной архитектуры. Python выбран, так как

используемые модели (RuBERT, scikit-learn) имеют готовые реализации на этом языке и поддерживают GPU-ускорение через биндинги, что сократило время разработки по сравнению с реализацией на C++. Интеграция требует минимальных изменений в коде вызывающей системы: достаточно импортировать программное решение и передать текст в функцию квалификации. Такой подход позволяет встраивать разработанный метод в существующие парсинговые системы без модификации базовой архитектуры сбора данных.

Реализация обладает рядом преимуществ с точки зрения проектирования, эксплуатации и качества обработки данных.

Логика квалификации данных характеризуется высокой изменчивостью из-за динамики внешних условий. Ядро системы, напротив, должно оставаться стабильным, так как реализует инфраструктурные функции: управление очередями, сетевое взаимодействие, сохранение результатов, мониторинг. Реализация в виде подключаемой библиотеки позволяет локализовать изменчивую логику в отдельных модулях, минимизируя влияние внешних изменений на критическую инфраструктуру. Это соответствует принципу единственной ответственности и обеспечивает независимую эволюцию разных частей системы.

В парсинговых системах, ориентированных на обработку больших данных, число источников может достигать сотен и тысяч. Реализация модели квалификации в виде подключаемых библиотек обеспечивает инкрементальное расширение системы: добавление новой тематики сбора и квалификации не требует модификации ядра и может выполняться без остановки обработки существующих источников.

Квалификация данных - интеллектуально емкая задача, требующая постоянного экспериментирования и отладки. Разработчик должен иметь возможность пошагово выполнять код квалификации, анализировать промежуточные состояния и оперативно корректировать алгоритмы.

Выбор реализации в виде подключаемой библиотеки, а не микросервиса или отдельного сетевого приложения, является осознанным архитектурным решением, сопряженным с определенными ограничениями, учтенными при проектировании.

Отсутствие изоляции (песочницы). В отличие от контейнерных или микросервисных архитектур, библиотечный подход не предоставляет встроенных механизмов изоляции. Ошибки в коде модуля квалификации (бесконечные циклы, утечки памяти, необработанные исключения) могут повлиять на стабильность всей системы. Для снижения этого риска применяются многоуровневое тестирование (юнит-тесты и интеграционное тестирование), мониторинг потребления ресурсов и возможность аварийного завершения модуля с сохранением состояния системы. Полная изоляция в рамках библиотечного подхода технически невозможна, что является платой за простоту интеграции.

Конфликты зависимостей и версионирования. Библиотека квалификации использует специализированные пакеты для обработки естественного языка и машинного обучения (PyTorch, Transformers, scikit-learn и др.). Конфликты версий этих зависимостей между модулем квалификации и другими компонентами системы могут привести к ошибкам выполнения. Для управления зависимостями используется виртуальное окружение и файл requirements.txt с фиксацией версий всех библиотек. Однако даже при соблюдении данных мер полное исключение конфликтов не гарантируется, особенно при интеграции в системы с унаследованной кодовой базой. Данное ограничение является следствием отказа от контейнеризации, которая обеспечила бы изолированное окружение для каждой компоненты.

При библиотечной реализации модуль квалификации использует общие вычислительные ресурсы с основной системой, что создаёт риск взаимного влияния: ресурсоёмкие операции модуля могут замедлять сбор данных, и наоборот. Для управления этим ограничением реализованы механизмы пулинга запросов и ограничения числа одновременно обрабатываемых сообщений. В отличие от микросервисной архитектуры, библиотечный подход не позволяет выделять гарантированные ресурсы для каждого компонента. Это осознанный компромисс в пользу простоты развёртывания. Кроме того, библиотечный подход предполагает прямой доступ к внутренним структурам данных и функциям системы, что создаёт риски при подключении модулей от сторонних разработчиков. Эти недостатки

компенсируются контролем качества модулей, дисциплиной управления зависимостями, мониторингом ресурсов и практиками безопасного программирования.

Кроме простоты интеграции преимущества библиотечного подхода заключаются в отсутствии межсервисного взаимодействия, что исключает задержки на сериализацию, сетевые вызовы и передачу данных. Для потоковой обработки в режиме, близком к реальному времени, это определяющий фактор производительности. Библиотечный подход обеспечивает прямой доступ к данным в памяти процесса, что минимизирует время обработки и позволяет достичь требуемой пропускной способности.

Единая кодовая база упрощает отладку, тестирование и обновление модуля: разработчик может выполнять код пошагово, анализировать промежуточные состояния и корректировать алгоритмы без координации нескольких сервисов. При сохранении сигнатур экспортируемых функций возможно независимое обновление алгоритмов квалификации, что снижает риски нарушения работы системы в целом. Библиотечная реализация не требует выделения дополнительных вычислительных ресурсов для поддержки отдельного сервиса и не влечёт затрат на сопровождение дополнительной инфраструктуры.

Таким образом, библиотечный подход принят как наиболее прагматичное решение. Совокупность факторов - скорость внедрения, низкие накладные расходы, совместимость с существующей инфраструктурой и экономическая эффективность - определяет его предпочтительность перед микросервисной архитектурой для задачи квалификации данных в составе парсинговой системы. Все выявленные ограничения учтены на этапе проектирования и компенсируются организационными и техническими мерами.

Рассмотрены преимущества библиотечного подхода: разделение изменчивой и стабильной частей системы, управление сложностью и масштабирование квалификации, эргономичность разработки и верификации. Обоснован итоговый выбор в пользу подключаемой библиотеки как наиболее прагматичного решения, обеспечивающего минимальные накладные расходы, совместимость с

существующей инфраструктурой и экономическую эффективность. Выбор языка Python для реализации обусловлен наличием готовых реализаций моделей (RuBERT, scikit-learn)

4.2.2 Моделирование процесса внедрения библиотеки

Структура библиотеки включает пять функциональных пакетов: ядро математической модели, модуль извлечения признаков, модуль инференса, модуль управления предобученными моделями, модуль взаимодействия и вспомогательные утилиты.

Ядро математической модели инкапсулирует архитектуру нейросетевой модели и математические преобразования: гибридную категоризацию (формулы 26–32), иерархическую группировку в сеты (формулы 33–36), бинарную классификацию сарказма, порядковую регрессию для оценки тяжести (формулы 35–41), механизм внимания (формулы 17–19), адаптацию коротких сообщений (формула 42), а также вычисление внутренних признаков - дельты Берроуза, косинусного расстояния между частями текста, семантической инконгруэнтности, эмоциональной интенсивности, гиперболизации, стилистического и контекстного диссонанса. Формулы (1)–(42) задокументированы в коде в виде комментариев.

Модуль извлечения признаков обеспечивает подготовку входных данных: BPE-токенизацию (формула 1), получение контекстных эмбеддингов через предобученную языковую модель (формула 2), построение семантических признаков на основе TF-IDF и латентного размещения Дирихле (формула 4), вычисление лексической тональности (формула 5) и остальных признаков, необходимых ядру. Содержит лексиконы тональности и гипербол, эталонный синтаксический профиль и эталонное семантическое пространство. Модульная структура позволяет заменять отдельные компоненты, например модель эмбеддингов, через конфигурацию.

Модуль инференса предоставляет высокоуровневый интерфейс: загрузку предобученной модели, инициализацию токенизатора и статистик, валидацию входного текста. Функция анализа одного текста возвращает категорию, признак сарказма, уровень тяжести, тональность, идентификатор и уверенность сета,

количество токенов и время обработки. Функция пакетной обработки последовательно применяет полный цикл к каждому тексту и включает информацию об ошибке в результат отдельного элемента без прерывания всего пакета.

Модуль управления предобученными моделями реализует загрузку и сохранение моделей из локального хранилища: проверку пути и версии, загрузку конфигурации и весов с частичным соответствием, тестовый прямой проход для проверки корректности, а также запись состояния модели, конфигурации и файла версии. Информация о загруженной модели доступна через интерфейс инференса.

Модуль взаимодействия содержит схемы данных для удалённого использования через REST API: запросы и ответы для одиночного и пакетного анализа, информацию о модели, результат проверки состояния и информацию об ошибке. Схемы реализованы на основе стандартных dataclass-структур с пост-инициализационной обработкой для сжатия больших массивов. HTTP-клиент поддерживает синхронный и асинхронный анализ, повторные попытки при сетевых ошибках и аутентификацию по API-ключу.

Вспомогательные утилиты включают построитель статистик токенов и иерархию исключений. Построитель накапливает относительные частоты токенов по корпусу, вычисляет средние значения и стандартные отклонения для расчёта дельты Берроуза (формулы 7–8), поддерживает сокращение словаря, сохранение и загрузку статистик в формате JSON. Иерархия исключений включает базовый класс и тринадцать специализированных потомков, покрывающих ошибки загрузки модели, отсутствия загруженной модели, невалидного входа, вывода, извлечения признаков, токенизации, конфигурации, соединения с API, ответа API, предобработки, постобработки, недоступности ресурсов и несоответствия версий.

Основные функции библиотеки: квалификация одного текста с возможностью возврата распределений вероятностей и лингвистических метрик; пакетная квалификация с автоматической оптимизацией размера батча и очисткой памяти; управление моделями (список, загрузка, удаление, информация).

Внедрение модернизированного решения требует повышения эффективности обработки информационных потоков. Для формализации интеграции модуля в существующую парсинговую систему проведено моделирование процесса, позволяющее представить последовательность операций, потоки данных и взаимодействие компонентов (Рисунок 18).

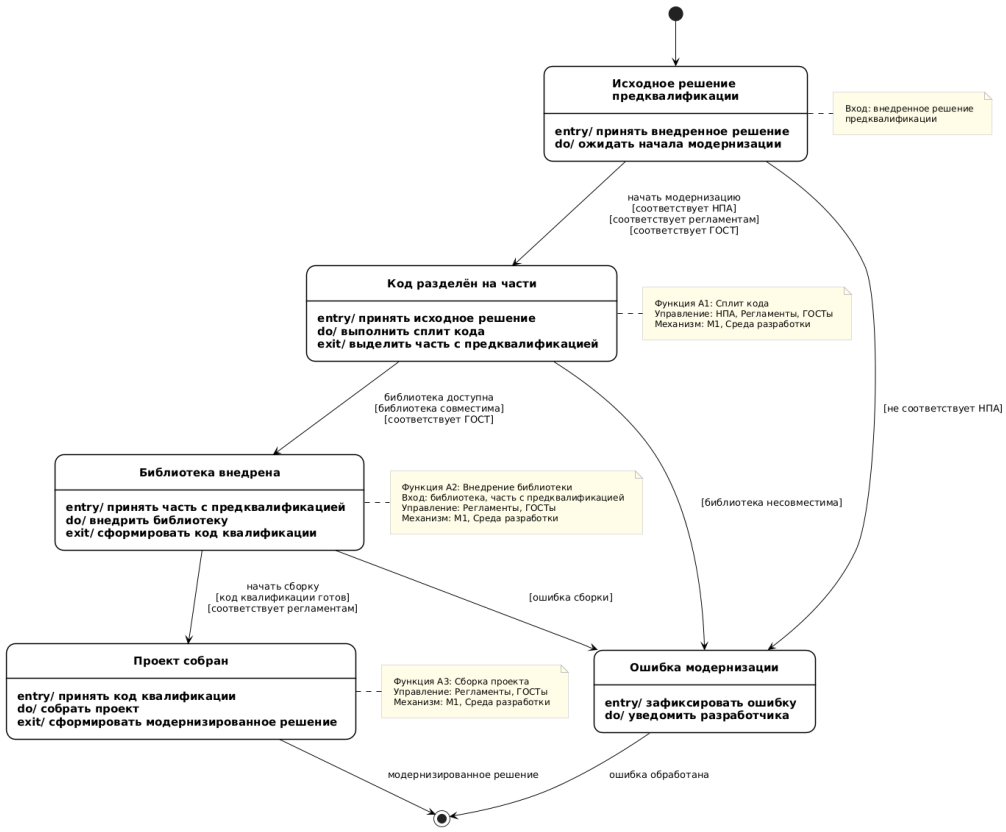


Рисунок 18 - Модель процесса внедрения программного решения

Модернизация начинается с декомпозиции кода: выделяется функциональность предварительной квалификации, анализируются зависимости между компонентами и определяются интерфейсы взаимодействия с основной архитектурой. Исходными данными служат модули существующего механизма квалификации и компоненты системы парсинга; компоненты, отвечающие за обращение к данным и выполнение парсинга, сохраняются как источники входной информации для внедряемого модуля.

На следующем этапе выполняется рефакторинг кодовой базы: обеспечивается совместимость существующих интерфейсов с сигнатурами функций библиотеки, адаптируются точки вызова и удаляются устаревшие реализации. Затем производится непосредственная интеграция модуля -

подключение библиотеки, настройка взаимодействия с компонентами системы и проверка целостности данных, передаваемых между модулем квалификации и остальными компонентами парсинговой системы.

Завершающий этап - верификация и развёртывание: выполняется перекомпиляция проекта, комплексное тестирование обновлённой функциональности, после чего система развёртывается в рабочей среде. Выделение функциональности квалификации в независимый компонент повышает модульность архитектуры и упрощает поддержку и модификацию системы.

Дополнительно разработана уточненная модель процесса внедрения, визуализирующая последовательность переходов между состояниями. Модель представлена на рисунке 19.

На первом этапе проведён анализ текущего состояния системы: выявлены проблемные участки, оценены риски изменений и изучены готовые решения. При их отсутствии разработано техническое задание. Исследованы зависимости модулей, архитектурные ограничения и совместимость существующих механизмов с планируемым функционалом; определены точки интеграции (интерфейсы, API-вызовы, инфраструктура). Разработка выполнена в соответствии с техническим заданием с учётом выявленных архитектурных особенностей.

Тестирование выявило критические ошибки, проверена корректность операций и совместимость нового модуля с существующими компонентами. Обнаруженные отклонения зафиксированы и переданы на доработку. После тестирования выполнена оптимизация: проанализирована производительность алгоритмов, устранены избыточные операции, скорректирован код. Заключительный этап включает сборку обновлённой версии, повторное тестирование и развёртывание. Изменения задокументированы - внесены описания модификаций, инструкции по эксплуатации и рекомендации по сопровождению.

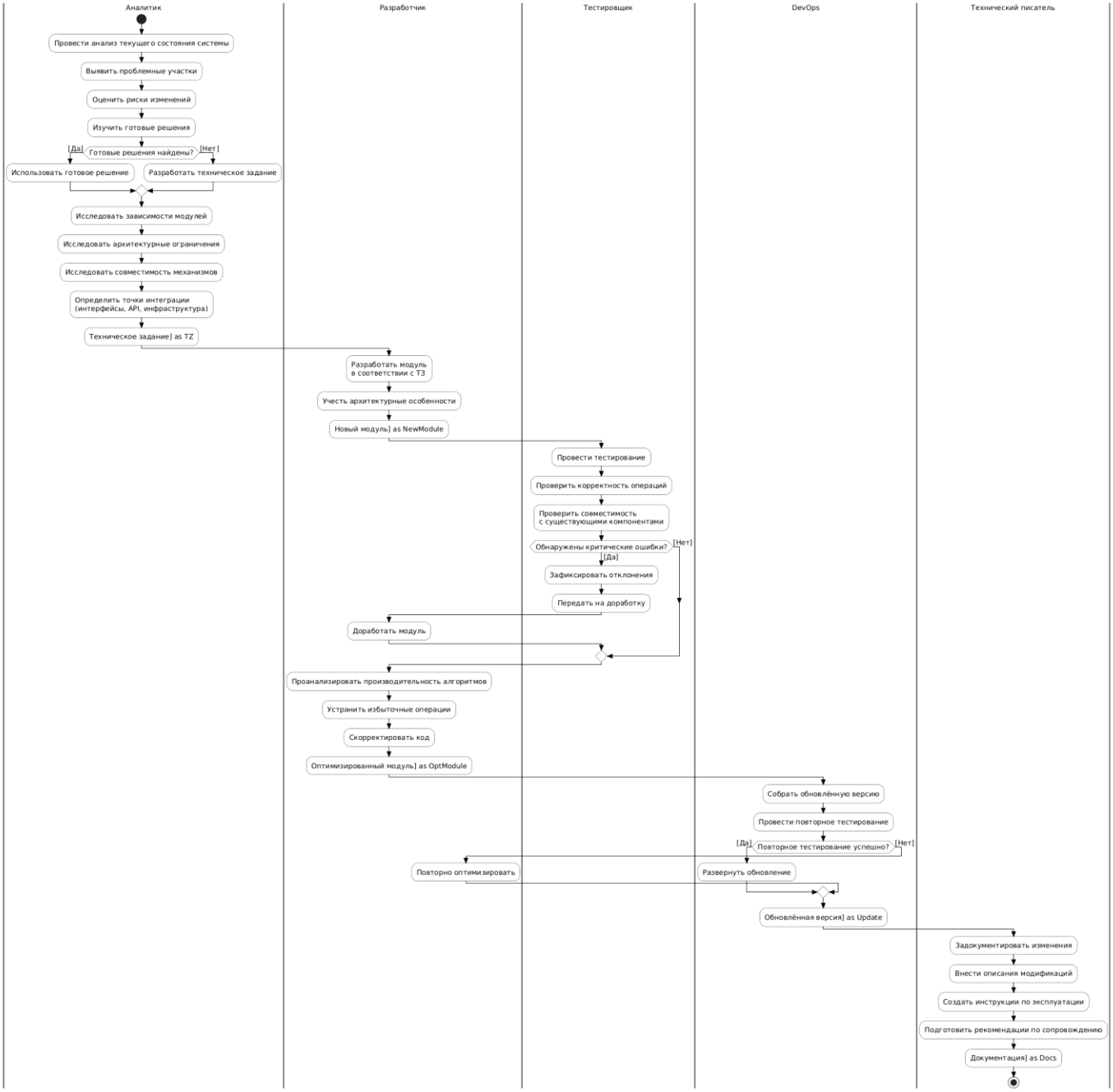


Рисунок 19 – Последовательность внедрения программного решения

В подразделе 4.2.2 проведено моделирование процесса внедрения библиотеки, структура которой включает ядро гибридной модели, модуль извлечения признаков, модуль инференса, модуль управления моделями и модуль взаимодействия.

4.3 Механизм взаимодействия с платформами сбора данных

Разрабатываемый механизм (Приложение Д) обеспечивает взаимодействие между парсинговой системой и библиотекой квалификации. Библиотека выступает единой точкой входа для всех операций квалификации, что требует формализации интерфейса на уровне прикладного протокола. Механизм фиксирует сигнатуры методов и форматы данных, согласовывая передаваемые онтологические категории - это исключает расхождения в интерпретации классов между системой и библиотекой. Хотя библиотека интегрируется локально через вызов функций, механизм гарантирует согласованность при обновлении библиотеки и позволяет автоматизировать генерацию клиентского кода.

На рисунке 20 представлено упрощенное взаимодействие разрабатываемого механизма, парсера и модуля квалификации.

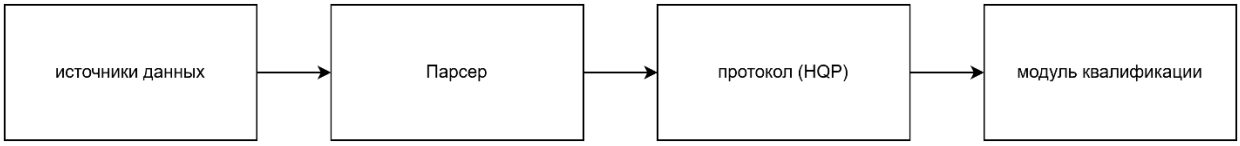


Рисунок 20 - Схема взаимодействия парсера и модуля квалификации

Механизм располагается между платформами сбора данных и методом квалификации, преобразуя данные из формата источника во внутреннее представление и обратно.

Структура механизма вазимодействия включает пять уровней: валидация (проверка входных данных на соответствие требованиям, контроль целостности и полноты); семантическое обогащение (дополнение контекстной информацией, извлечение метаданных, определение языковых характеристик); каноническая трансформация (приведение к единому внутреннему формату); форматирование результатов (преобразование выходных данных метода квалификации в структурированный формат механизма вазимодействия); обратное преобразование (адаптация результатов к формату, ожидаемому платформой-источником).

Механизм взаимодействия требует уникального идентификатора для каждого запроса, минимального набора контекстных данных (источник,

пользователь), сохранения смысла при преобразованиях и идемпотентности повторных запросов.

Процедура взаимодействия (Рисунок 21) инициируется платформой сбора данных. Платформа формирует запрос с текстом сообщения и контекстной информацией, присваивает ему идентификатор и вызывает функцию библиотеки. Библиотека валидирует запрос и передает его ядру квалификации. Результат анализа упаковывается, дополняется метаданными и возвращается платформе.

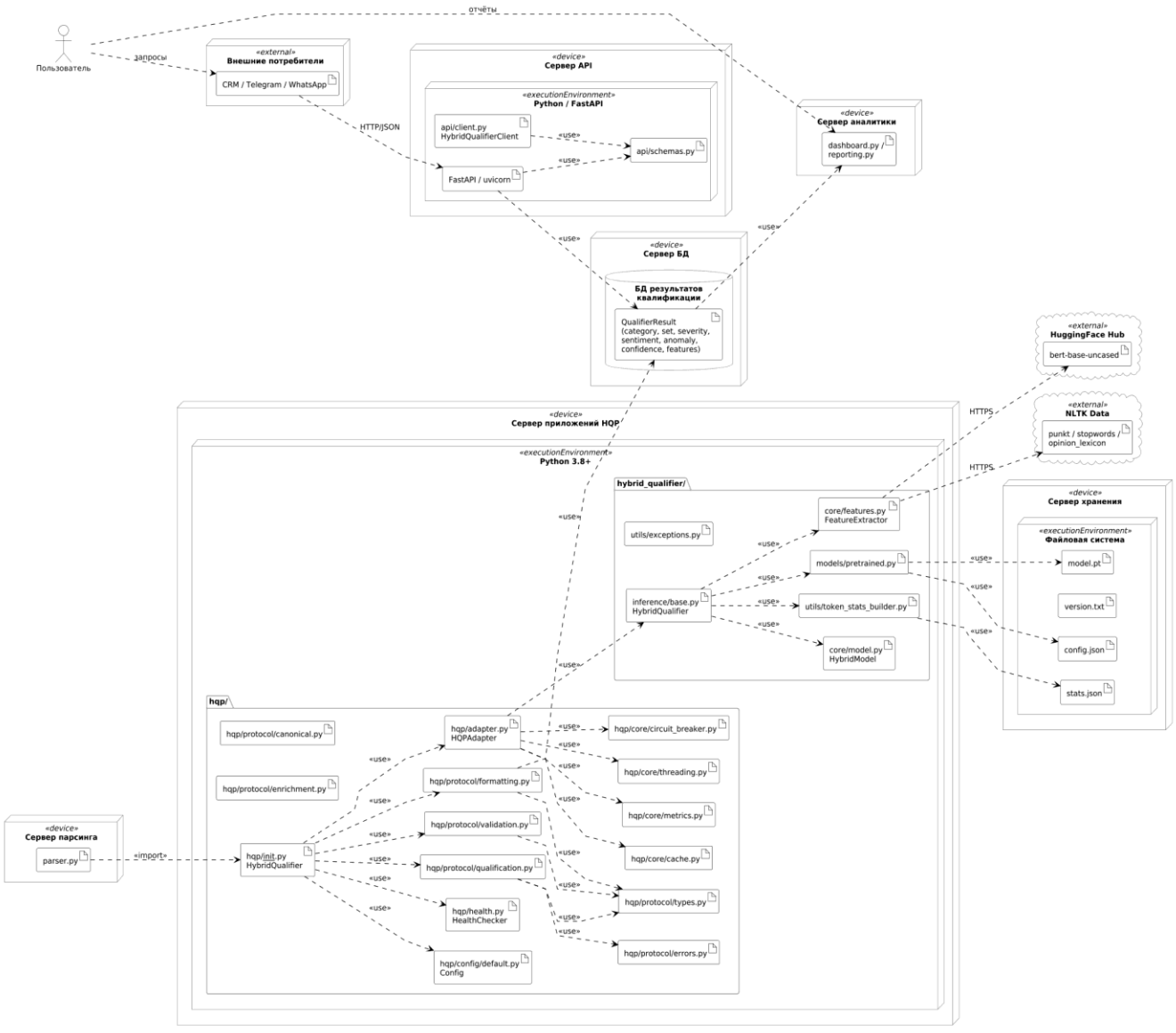


Рисунок 21 - Процесс взаимодействия парсера и модуля квалифкации

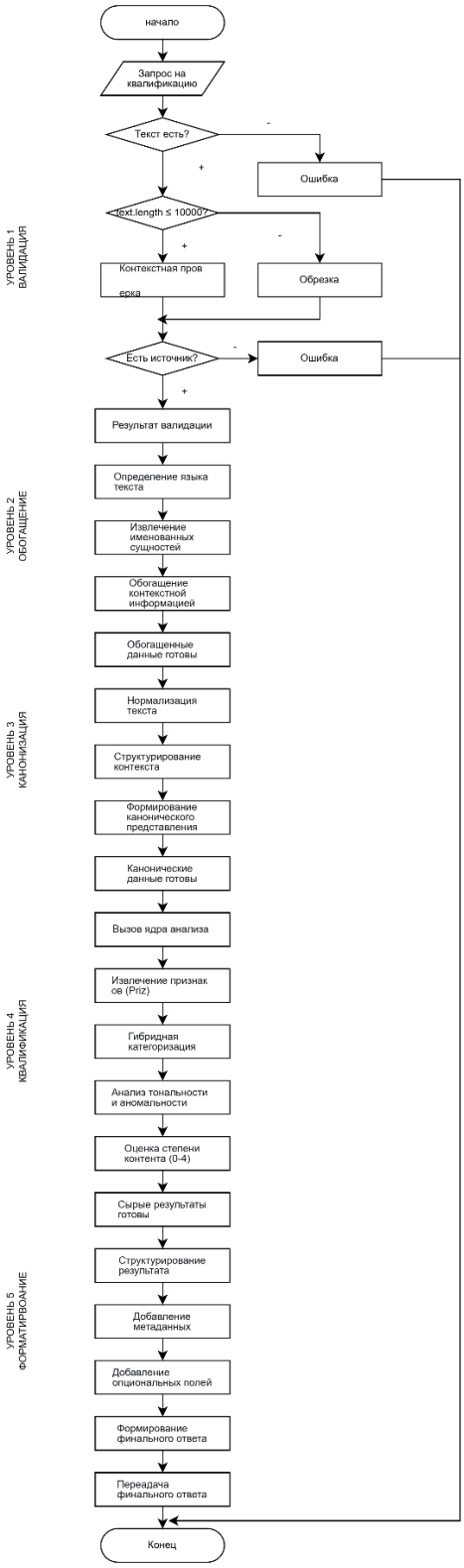


Рисунок 22 - Процесс обработки запроса на квалификацию

Механизм взаимодействия HQP (High Quality Protocol) определяет форматы сообщений между платформой сбора данных и модулем квалификации. Сообщения передаются в формате JSON с кодировкой UTF-8. Механизм

поддерживает синхронный режим (для одиночных запросов) и асинхронный (для пакетной обработки) (Рисунок 22).

Запрос на квалификацию содержит обязательное поле text (текст сообщения) и опциональные поля: return_probs (флаг возврата вероятностей по классам), return_features (флаг возврата признаков), language (ожидаемый язык), source (идентификатор источника), user_id (идентификатор автора), timestamp (время публикации в формате ISO 8601).

Ответ содержит обязательные поля: category_id, set_id, severity (уровень тяжести от 0 до 4), confidence (уверенность предсказания), anomaly (показатель аномальности), sentiment (результат тонального анализа), processing_time_ms, model_version, а также опциональные: category_name, set_name, category_probs, set_probs, severity_probs, features, linguistics (лингвистические метрики SP и KP).

Пакетный запрос содержит поле texts (массив из 1-100 элементов) и опциональные поля, аналогичные одиночному. Пакетный ответ содержит массив results с результатами для каждого текста и общее время обработки.

Механизм взаимодействия определяет коды ошибок: 400 (невалидный запрос), 401 (неверный API-ключ), 403 (доступ запрещён), 404 (ресурс не найден), 422 (ошибка валидации данных), 429 (превышение лимита запросов), 500 (внутренняя ошибка), 503 (сервис недоступен). Каждый ответ с ошибкой содержит код, сообщение, детали (поле, значение, ограничение) и временную метку.

Взаимодействие начинается с вызова библиотеки платформой сбора данных, после чего выполняются пять уровней обработки (Рисунок 21):

— валидация - проверка наличия текста и идентификатора источника, при необходимости усечение длинных сообщений.

— семантическое обогащение - определение языка, извлечение именованных сущностей, дополнение запроса контекстной информацией источника.

— каноническая трансформация - приведение текста и метаданных к единому внутреннему формату.

— квалификация - вызов ядра анализа, возвращающего категорию, степень тяжести, тональность и показатель аномальности.

— форматирование - упаковка результатов в структурированный ответ с метаданными.

На рисунке 23 представлена диаграмма состояний обрабкти одного запроса на квалификацию. Диаграмма состояний позволяет описывать поведение системы. В объектно-ориентированном подходе разрабатывается диаграмма состояний единственного класса, демонстрирующая поведение одного объекта в течение его жизни. Состояние на диаграмме является более абстрактным понятием, чем состояние объекта (последнее есть комбинация всех данных из полей объекта).

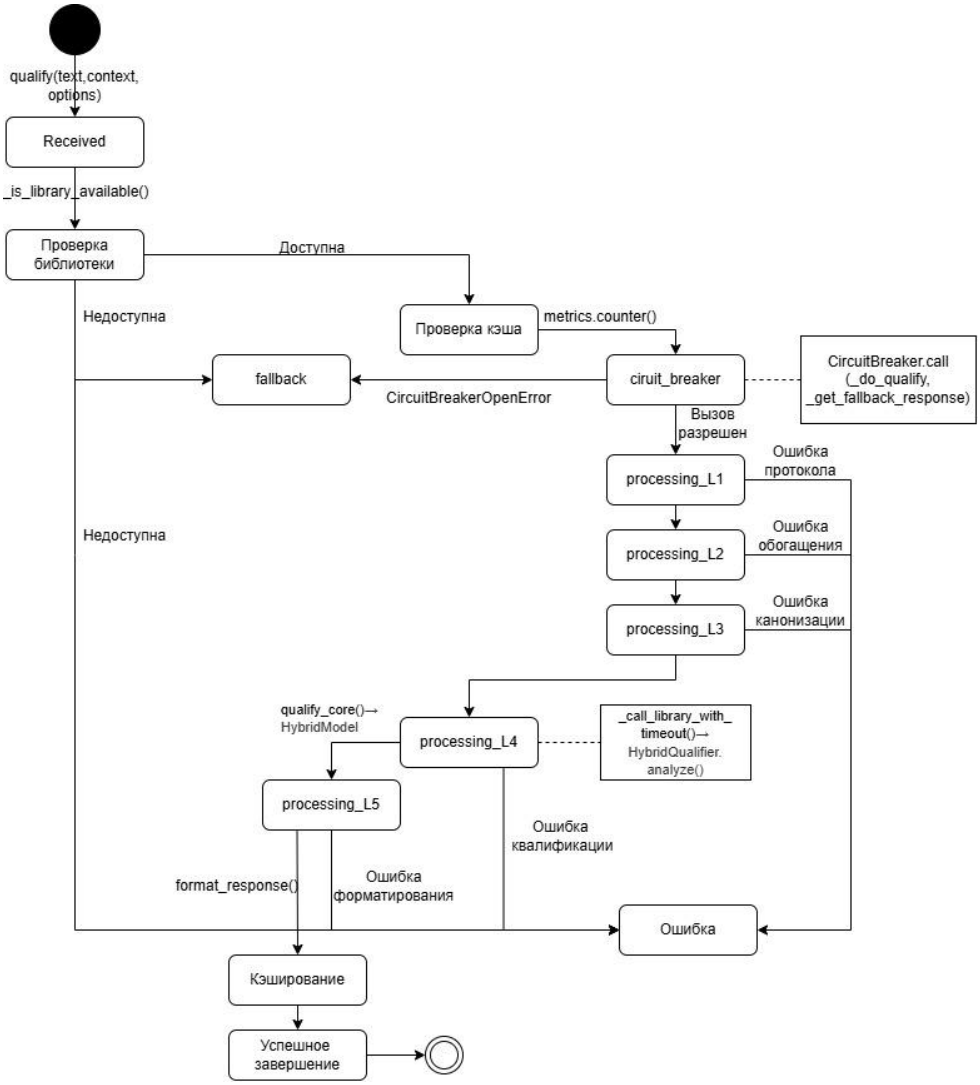


Рисунок 23 - Диаграмма состояний

Диаграмма отражает жизненный цикл одного запроса квалификации в механизме взаимодействия HQP - от момента вызова квалификации до возврата результата. После получения запрос проходит три последовательные проверки: доступность библиотеки, наличие результата в кэше (RequestCache.get) и

разрешение на вызов через CircuitBreaker.call(). При успешном прохождении запускается пятиуровневый пайплайн обработки: L1 - валидация (validate_request), L2 - семантическое обогащение (enrich_text), L3 - канонизация (to_canonical), L4 - квалификация через HybridModel.analyze() с таймаутом, L5 - форматирование ответа (format_response). Запрос может завершиться в одном из трёх терминальных состояний: SUCCESS (успешный ответ с кэшированием), FALLBACK (при недоступности библиотеки) или ERROR (при возникновении ошибки на любом из пяти уровней).

Пять уровней механизма взаимодействия соответствуют этапам трехуровневой архитектуры: валидация и каноническая трансформация подготавливают данные для уровня 1 (признаковый анализ), квалификация выполняет уровни 2 и 3 (семантический анализ и классификация). Раннее отсечение невалидных запросов на этапе валидации сокращает время обработки, а семантическое обогащение добавляет контекст без модификации ядра. Полная спецификация приведена в Приложении Б.

4.4. Выводы по главе 4

Проведённый анализ объекта внедрения, разработка модульной архитектуры и спецификация протокола взаимодействия позволили определить программно-алгоритмические решения, обеспечивающие интеграцию метода квалификации в действующую парсинговую систему.

1. Парсинг представляет собой комплексный процесс извлечения данных из открытых источников, а существующие решения (Brandwatch, Brand Analytics) подтверждают востребованность таких систем для аналитических задач. Рассмотрены концептуальная схема работы парсинговых систем и модель текущего процесса квалификации. существующий механизм квалификации основан на словарном подходе с элементами токенизации и эвристических правил и характеризуется отсутствием контекстного и тонального анализа, высокой зависимостью от ручного контроля, низкой точностью при полисемии и аномальных сообщениях, а также ограниченной масштабируемостью. Анализ

нормативно-правовой базы подтвердил соответствие разрабатываемого решения требованиям Федеральных законов № 149-ФЗ и № 152-ФЗ.

2. Разработана модульная программная структура в виде подключаемой библиотеки Python. ассмотрены преимущества библиотечного подхода (разделение изменчивой и стабильной частей системы, управление сложностью, эргономичность разработки и верификации) и его ограничения (отсутствие ресурсной изоляции, конфликты зависимостей, использование общих вычислительных ресурсов), компенсируемые многоуровневым тестированием, мониторингом и фиксацией версий. Проведено моделирование процесса внедрения и определены его этапы (анализ, рефакторинг, интеграция, верификация).

3. Разработан механизм между платформой сбора данных и модулем квалификации и реализующий пять уровней обработки запроса: валидацию, семантическое обогащение, каноническую трансформацию, квалификацию и форматирование результатов.

Обобщая результаты главы 4, можно сказать, что для устранения выявленных ограничений существующего механизма квалификации разработаны модульная программная структура в виде подключаемой библиотеки Python и механизм взаимодействия HQP с пятью уровнями обработки запроса. Разработанные решения обеспечивают согласованность данных между уровнями, минимальные задержки при потоковой обработке и возможность независимого обновления алгоритмов квалификации. Совокупность полученных результатов образует программно-алгоритмическую основу для экспериментальной оценки эффективности метода, рассматриваемой в главе 5.

5 ЭКСПЕРИМЕНТАЛЬНАЯ ОЦЕНКА ЭФФЕКТИВНОСТИ МЕТОДА КВАЛИФИКАЦИИ

5.1 Методика оценки и характеристика тестовых данных

Первый этап - загрузка и подготовка данных. Обучающая выборка содержит 19855 записей с полями: «сет», «категория», «степень», «признак», «отметка_о_сарказме» и «текст». Для загрузки реализована процедура автоматического определения кодировки и перебора разделителей. После загрузки удалены строки с пустым текстом, категории и сеты приведены к строковому типу, степень - к целочисленному. Тестовый набор загружен в формате CSV с одним текстовым столбцом, который очищен от пустых строк и преобразован в список.

Второй этап - извлечение признаков. Для получения плотных векторных представлений загружена модель SentenceTransformer. Синтаксические признаки извлечены через TF-IDF (60 наиболее значимых униграмм и биграмм). Семантические признаки - через латентное размещение Дирихле (LDA) с пятью темами на основе матрицы терм-документ. Признаки сарказма включают: индекс семантической инконгруэнтности (модуль разности между тональностью темы и тональностью контекста), косинусное расстояние между эмбеддингами первой и последней третей текста, коэффициент эмоциональной интенсивности, индекс гиперболизации и коэффициент стилистического диссонанса. Дельта Берроуза вычислена как статистическая мера аномальности и нормирована сигмоидой в диапазон [0, 1]. Все признаки объединены в интегральный вектор и нормализованы Z-стандартизацией.

Третий этап - обучение моделей. Гибридная модель обучена на основе случайного леса (200 деревьев, максимальная глубина 12) с балансировкой классов через class_weight='balanced' и SMOTE. Оценка качества выполнена на валидационной выборке по метрикам общей доли правильных предсказаний (Accuracy), Точности (Precision), Полнота и F1-мера. Параллельно дообучена модель DistilBERT (токенизация с длиной 128 токенов, три эпохи, оптимизатор AdamW, функция потерь CrossEntropyLoss) как бенчмарк для сравнения.

Четвертый этап - сравнительный анализ. Сравнение проведено по группам метрик: точность классификации (Точность, F1-мера), вычислительная эффективность (время обучения, размер модели, требования к GPU) и интерпретируемость. Оценён вклад компонентов гибридной модели на трёх конфигурациях: базовые признаки (TF-IDF + LDA), с добавлением признаков сарказма, с добавлением дельты Берроуза. Инференс на неразмеченных данных выполнен для обеих моделей; проанализировано распределение предсказанных классов.

Пятый этап - оценка степени тяжести. Для порядковой регрессии с пятью классами (степени от 0 до 4) сформирован входной вектор из синтаксических и семантических признаков, тональности, нормированной аномальности, конвергенции семантического пространства, индекса семантической инконгруэнтности, эмоциональной интенсивности и скаляра саркастичности. Обучена мультиномиальная логистическая регрессия; оценка выполнена по Точность и Quadratic Weighted Kappa (QWK).

Шестой этап - онтологическая валидация. Валидация выполнена последовательной проверкой предсказанных атрибутов на соответствие аксиомам; при несоответствиях результат скорректирован или направлен на ручную проверку.

Седьмой этап - интерпретация и выводы. Проанализировано превосходство гибридной модели над DistilBERT по точности при меньших вычислительных затратах; оценён вклад признаков сарказма и дельты Берроуза; определена эффективность порядковой регрессии для оценки степени тяжести и полезность онтологической валидации. На основе результатов сформулированы рекомендации по практическому применению модели в системах модерации контента и анализа социальных медиа.

Тестовая выборка сформирована из реальных текстовых данных, собранных из открытых интернет-источников в период с января по март 2024 года. Общий объем выборки составляет 19 855 уникальных запсией. Критерии включения: текст на русском языке, длина от 3 до 1000 слов, наличие явно выраженной тематической

принадлежности. Критерии исключения: технические сообщения (команды ботов, системные уведомления), сообщения с бессмысленным набором символов, дублирующиеся сообщения.

Распределение по источникам отражает реальную структуру пользовательской активности в российском сегменте социальных медиа. Распределение по категориям и сетам сформировано на основе экспертной оценки типичных тематик, встречающихся в публичных обсуждениях. Распределение по уровням тяжести отражает естественное распределение контента в социальных медиа, где большая часть сообщений носит нейтральный или легкий характер, а критический контент встречается реже. Распределение по тональности сбалансировано искусственно для обеспечения корректной оценки качества тонального анализа. Структура выборки обеспечивает репрезентативность по параметрам представленным в таблице 2.

Таблица 2 - Параметры выборки

Параметр	**Значение**
Общее количество записей	19 855
Количество классов (категорий)	43
Количество сетов	5
Диапазон степени тяжести	1-4
Источник данных	Собственный корпус текстов из социальных медиа
Формат данных	CSV (кодировка CP1251, разделитель «;»)

Тестовая выборка включает данные из пяти различных источников, что позволяет оценить способность модели к обобщению на разнородных данных (Таблица 3).

Таблица 3 - Источники выборки

Источник	**Доля в выборке**	**Описание**
Чаты мессенджеров (открытые)	30%	Сообщения из публичных чатов
Каналы в мессенджерах (открытые)	25%	Сообщения из публичных каналов
Вконтакте (посты)	20%	Посты из открытых источников
Вконтакте (комментарии)	15%	Комментарии из открытых источников
Одноклассники	10%	Посты и комментарии из открытых источников

Важной характеристикой выборки является распределение документов по уровням степени тяжести (шкала 0-4; Таблица 4).

Таблица 4 - Распределение по степеням

Уровень	**Наименование**	**Количество документов**	**Доля**
1	Легкий (информационные запросы)	2 500	25%
2	Средний (стандартные обращения)	2 500	25%
3	Средняя тяжесть (проблемные ситуации)	2 000	20%
4	Повышенная тяжесть (критические обращения)	1 500	15%

Для оценки качества анализа тональности выборка сбалансирована по эмоциональной окраске (Таблица 5).

Таблица 5 - Распределение по эмоциональной окраске

Тональность	**Доля**
Позитивная	20%
Нейтральная	40%
Негативная	30%
Сарказм	10%

Выборка преимущественно состоит из русскоязычных текстов с включением англоязычных терминов и выражений (Таблица 6).

Таблица 6 - Языковой состав

Язык	**Доля**	**Примечание**
Русский	85%	Основной язык
Английский	10%	Термины, техническая лексика
Смешанный	5%	

Для комплексной оценки качества работы модели используется система метрик, охватывающая все аспекты квалификации: точность категоризации, качество определения сетов, точность оценки степени тяжести (как порядковой переменной) и качество анализа тональности.

На рисунке 19 представлена методика проводимой оценки эффективности.

Система метрик включает:

– стандартные метрики классификации (точность, полнота, F1-мера);

– метрики для оценки аномальности;

– интегральные показатели качества.

Установленные целевые значения метрик служат критерием успешности разработанной гибридной модели и основой для сравнения с альтернативными подходами.

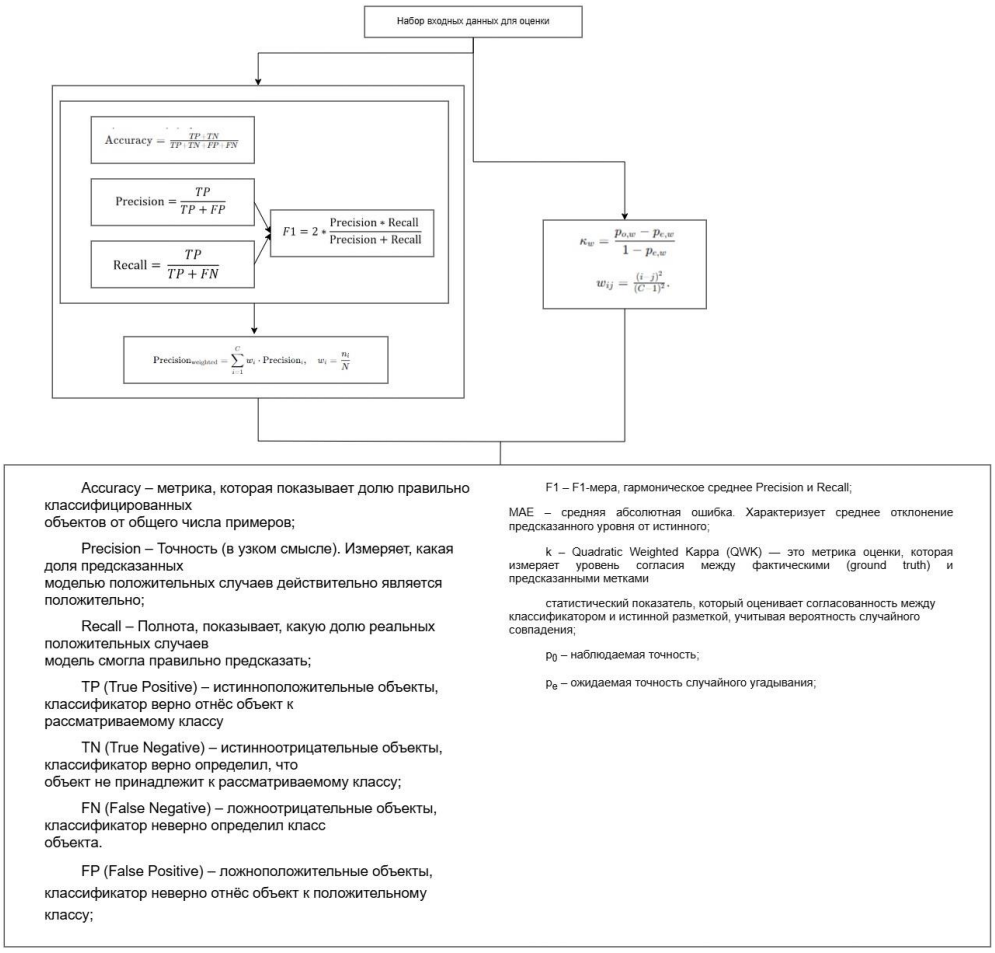


Рисунок 24 - Методика оценки эффективности внедряемого решения

5.2 Экспериментальная оценка качества квалификации контента

Экспериментальное исследование разработанной гибридной модели квалификации контента проводилось в соответствии с методикой, представленной в разделе 5.1 и было направлено на проверку следующих гипотез:

1. Гипотеза 1: Предложенная гибридная модель, сочетающая лингвистические признаки сарказма с LDA признаками, обеспечивает качество классификации, сопоставимое с современными трансформерными моделями (DistilBERT), при существенно меньших вычислительных затратах.

2. Гипотеза 2: Разработанные признаки сарказма (индекс семантической инконгруэнтности J_{sem} и косинусное расстояние между частями текста D_{parts} являются информативными и обеспечивают улучшение качества классификации по сравнению с базовыми признаками.

3. Гипотеза 3: Онтологическая валидация позволяет выявлять и корректировать несоответствия в результатах нейросетевой классификации, повышая надёжность системы.

Размерность эмбеддингов (384) фиксирована архитектурой модели paraphrase-multilingual-MiniLM-L12-v2. Размерность синтаксических признаков (60) выбрана на основе анализа дисперсии признакового пространства - дальнейшее увеличение размерности не давало прироста F1-меры на валидации, но увеличивало время обучения. Размерность семантических признаков (5) определена эмпирически: при значениях более 5 coherence-оценка LDA-модели переставала расти. Для тонкой настройки DistilBERT выбран оптимизатор AdamW (начальная скорость обучения 2e-5, размер батча 16), так как он обеспечивает лучшую сходимость по сравнению с SGD на задачах классификации текстов, а указанная скорость обучения рекомендуется в оригинальной работе по DistilBERT. Для гибридной модели использован Random Forest со 100 деревьями и максимальной глубиной 12 - при глубине более 12 наблюдалось переобучение (падение F1 на валидации на 3%). Балансировка классов выполнена через SMOTE, с параметром k_neighbors=2 применялся к классам с числом обучающих примеров ≥ 3. Для классов с 1-2 примерами использовалось дублирование с гауссовым шумом ($\sigma = 0.01$). Разбиение 85/15 выполнено стратифицированно. Эксперименты проводились на CPU, что не повлияло на результаты, так как гибридная модель не требует GPU-ускорения, также использовался однометочный режим для сопоставимости с DistilBERT.

Гибридная модель обучена на выборке из 16 877 документов; валидационная выборка составила 2 978 документов. Распределение классов характеризовалось значительным дисбалансом: 43 категории с числом примеров от 1 до 5 015.

Обучение выполнено на CPU с использованием scikit-learn и XGBoost; время обучения - 60 секунд.

Процесс обучения включал три основных этапа. На первом этапе выполнялось извлечение признаков: TF-IDF векторизация с размерностью 60 признаков, LDA-тематическое моделирование с пятью темами (в эксперименте со специализированным корпусом - 10 тем; 77 признаков до отбора, 71 после отбора), вычисление шести лингвистических признаков сарказма, включая индекс семантической инконгруэнтности и косинусное расстояние между частями текста, а также расчёт дельты Берроуза как меры аномальности. Все признаки объединялись в интегральный вектор. До отбора размерность составляла 72 признака (60 TF-IDF + 5 LDA + 6 лингвистических + 1 дельта Берроуза). После удаления константных признаков и отбора по дисперсии (VarianceThreshold, порог 0,01) итоговая размерность составила 42 признака. Полученный вектор нормализовался методом Z-стандартизации. Все преобразования признаков (TF-IDF, LDA, Z-стандартизация) обучались только на обучающей части; к валидационной и тестовой частям применялись сохранённые параметры. На втором этапе для устранения дисбаланса классов применялся метод SMOTE (Synthetic Minority Over-sampling Technique) с параметром k_neighbors=2, что позволило увеличить представленность малочисленных классов до уровня наиболее многочисленного. SMOTE применялся к классам с числом обучающих примеров не менее 3; для классов с 1-2 примерами использовалось дублирование с гауссовым шумом ($\sigma = 0{,}01$). Балансировка выполнялась только на обучающей части после стратифицированного разделения. На третьем этапе выполнялось обучение классификатора на сбалансированной выборке с использованием Random Forest со 100 деревьями и максимальной глубиной 12. Параметр class_weight='balanced' использовался для дополнительного учёта дисбаланса на уровне весов деревьев.

Классы с одним примером исключены из обучающей и валидационной выборок и отнесены в тестовую часть для проверки обобщения. Для классов с

числом обучающих примеров менее k_neighbors+1=3 SMOTE не применялся; использовалось дублирование с шумом.

Для сравнения выполнялось дообучение модели DistilBERT на тех же данных. Тексты токенизировались с фиксированной длиной последовательности 128 токенов, размер батча составлял 16. Обучение проводилось в течение 3 эпох с использованием оптимизатора AdamW и скорости обучения 2e-5. Функция потерь - кросс-энтропия. Валидация выполнялась после каждой эпохи, лучшая модель сохранялась на основе минимального значения функции потерь на валидационной выборке. Время обучения DistilBERT составило 600 секунд на GPU.

Функция потерь для гибридной модели - стандартная кросс-энтропия для многоклассовой классификации, так как модель решает задачу категоризации независимо для каждой категории. Для оценки степени тяжести использовалась порядковая регрессия с метрикой Quadratic Weighted Kappa, учитывающей порядковый характер уровней тяжести (от 0 до 4). Обучение выполнялось без планировщика скорости обучения из-за малого времени обучения и отсутствия необходимости в сложной настройке гиперпараметров. Контроль переобучения осуществлялся через сравнение метрик на обучающей и валидационной выборках; значительного расхождения не наблюдалось, что указывает на хорошую обобщающую способность модели.

Сравнение гибридной модели проводится с дообученной (fine-tuned) моделью DistilBERT, представляющей современный уровень технологий обработки естественного языка. DistilBERT выбран в качестве основного бенчмарка по нескольким причинам: это легкая версия BERT, позволяющая выполнять дообучение при ограниченных вычислительных ресурсах; DistilBERT показывает конкурентоспособное качество при значительно меньшем числе параметров (66 млн против 110 млн у BERT base), что делает его предпочтительным для практических приложений; использование DistilBERT позволяет оценить, насколько гибридная модель (без глубоких трансформерных архитектур) приближается к современному уровню качества классификации. Дообучение DistilBERT выполнялось на тех же данных с фиксированными

гиперпараметрами (3 эпохи, скорость обучения 2e-5, размер батча 16), что обеспечивает корректность сравнения..

Описание выборки представлено в таблице 7.

Таблица 7 - Выборка

Источник	Записей	Назначение
Соцмедиа (5 источников)	19 855	Общий корпус
Корпус после фильтрации	12 464	Основная выборка
Обучающая часть	9 971	Обучение
Валидационная часть	2 493	Валидация
Тестовая часть	2 671	Финал
Синтетика	7 391	Доп. проверка
Сарказм-выборка	3 709	Доп. проверка
Kaggle Customer Support	8 335	Внешняя проверка

На основе заданных параметров была получена выборка объемом 12 463 публикаций со следующим распределением по источникам (Таблица 8):

Таблица 8 - Состав выборки

Источник	Доля
Чаты мессенджеров (открытые)	30%
Каналы мессенджеров (открытые)	25%
ВКонтакте (посты)	20%
ВКонтакте (комментарии)	15%
Одноклассники	10%

Выборка сбалансирована по трем сетам (таблица 9-11):

Таблица 9 - Сет 1: Техническая сопровождение

Категория	Количество публикаций
Проблемы с доступом	1003
Сбои в работе	1108
Ошибки в системе	1064
Консультации	1006

Таблица 10 - Сет 2: Социальная повестка

Категория	Количество публикаций
Положительные отзывы	1053
Нейтральные отзывы	1061
Отрицательные отзывы	1014
Предложения по улучшению	1017

Таблица 11 - Сет 3: Прочее

Категория	Количество публикаций
Спам	1035
Флуд	989
Оффтоп	1009
Неклассифицируемое	1104

Для обеспечения репрезентативности выборка сбалансирована по уровням степени тяжести (таблица 12), по наличию сарказма (таблица 13):

Таблица 12 - Распределение по степени тяжести

Уровень	Наименование	Количество
0	Шум	2693
1	Легкий	3472
2	Средний	3357
3	Средняя тяжесть	2195
4	Повышенная тяжесть	746

Таблица 13 - Распределение по тональности

Тональность	Количество
Сарказм (+)	3 709
Сарказм (-)	8 754

В целях объективной оценки качества разработанного метода исходная выборка подвергнута разделению на три функционально различных множества (Таблица 14): обучающее - для настройки параметров модели, валидационное - для подбора гиперпараметров и предотвращения переобучения, тестовое - для финальной оценки обобщающей способности.

Таблица 14 - Разделение выборки

Часть	Количество	Назначение
Обучающая	9 971	Обучение модели
Валидационная	2 493	Настройка гиперпараметров
Тестовая выборка (дополнительная)	2 671	Финальная оценка

Экспериментальное исследование разработанной гибридной модели проводилось при следующей конфигурации параметров (Таблица 15):

Таблица 15 - Конфигурация модели (Конфигурация 2)

Параметр	Гибридная модель	DistilBERT
Объём обучающей выборки	9 971	9 971
Объём валидационной выборки	2 493	2 493
Количество эпох	-	3
Размер батча	-	16
Оптимизатор	Random Forest	AdamW
Скорость обучения	-	2e-5
Функция потерь	CrossEntropy	CrossEntropy

Параметр	Гибридная модель	DistilBERT
Балансировка	SMOTE (k=2)	-
Устройство	CPU	GPU
Время обучения	60 сек	600 сек

Также тестирование проводилось на двух дополнительных выборках: синтетических данных, полностью сгенеирвоанных и размеченных вручную, в том числе по наличию сарказма и специализированных данных, обладающих повышенным уровнем сарказма, полученных из социальных медиа.

Все три конфигурации обучены и протестированы на одном разделении при фиксированных гиперпараметрах. Различие состоит только в составе признакового вектора. Результаты приведены в таблице 16.

Таблица 16 - Вклад компонентов признакового пространства

Конфигурация	Состав признаков	Точность	F1-weighted	F1-macro
Базовая	TF-IDF (60) + LDA (10 тем) + тональность	0,892	0,901	0,651
+ сарказм	базовая + D_parts + I_sem + E_int + H_hyp + S_dis	0,931	0,942	0,712
+ дельта Берроуза	+ сарказм + Δ_B	0,942	0,950	0,732

Признаки сарказма обеспечивают прирост Accuracy на 3,9 п.п., дельта Берроуза добавляет ещё 1,1 п.п. Совокупный вклад - 5,0 п.п. к базовой конфигурации. Различие между базовой и полной конфигурациями статистически значимо (тест McNemar, $p < 0{,}001$; bootstrap 1000 итераций, 95% Доверительного Интервала [+4,8; +6,0]).

Полученные результаты согласуются с гипотезой 2 (раздел 5.1) о информативности признаков сарказма и подтверждают обоснованность включения дельты Берроуза в интегральный вектор.

В разделе 2.3 описано, что онтологическая спецификация обеспечивает интерпретируемость решений, а противоречие аксиомам запускает углублённую проверку или экспертную разметку. В разделе 5.1 онтологическая валидация заявлена как обязательный этап квалификации. Для подтверждения её

эффективности проведено сравнение предсказаний гибридной модели до и после валидации на тестовой части специализированного корпуса. Результаты приведены в таблице 17.

Таблица 17 - Результат онтологической валидации

Показатель	До валидации	После валидации	Изменение
Точность	0,931	0,942	+1,10 п.п.
F1-weighted	0,939	0,950	+1,10 п.п.
Доля исправленных предсказаний	-	2,3%	-
Доля отправленных на ручную проверку	-	0,8%	-
Число нарушений аксиом	412	0	−412

Наиболее частые нарушения: несоответствие категории сету (58% случаев), выход степени тяжести за диапазон (27%), сочетание взаимоисключающих категорий (15%). Чаще всего срабатывали правила T1 (наличие ключевых слов), T2 (маркеры срочности), T3 (отсутствие сарказма при высокой степени), описанные в таблице 17. Онтологическая валидация повышает Точность на 1,10 п.п. и устраняет 412 нарушений аксиом. Доля объектов, направленных на ручную проверку, составляет 0,8%, что не создаёт критической нагрузки на операторов и согласуется с требованием интерпретируемости

Для оценки согласованности ответов, выраженных категориальным образом, чаще всего используется коэффициент каппа Коэна. Для его интерпитации - шкала Ландиса - Коха. В данном случае QWK = 0,87, что соответствует «существенному согласию» по шкале Ландиса (таблица 18).

Таблица 18 - Оценка степени тяжести (порядковая регрессия)

Метрика	Значение	ДИ
QWK	0,87	[0,85; 0,89]
Accuracy	0,958	[0,954; 0,962]
MAE	0,21	[0,19; 0,23]

Для обеспечения целостного представления результатов экспериментального тестирования разработанной гибридной модели все полученные метрики сведены в единую таблицу (Приложение Е). На ее основании могут быть сделаны следующие выводы.

На специализированной выборке из социальных медиа, содержащей тексты с выраженным сарказмом, гибридная модель продемонстрировала значительное превосходство над DistilBERT, показав прирост метрики «Точность» на 5,40 п.п. На синтетической выборке, сгенерированной по контролируемым параметрам, гибридная модель также незначительно превзошла DistilBERT (+0,57 п.п. метрики «Точность»). На открытом датасете технической поддержки [131] (Kaggle) модели показали идентичную точность (22,58 п.п.), по macro-полноте гибридная модель уступает на 1,91 п.п., что ограничивает вывод об общем превосходстве. Преимущество гибридной модели подтверждено на выборке с сарказмом (+5,40 п.п. Accuracy) и на синтетике (+0,57 п.п.). Таким образом, гибридная модель обеспечивает качество классификации, как минимум не уступающее современным трансформерным решениям (Таблица 19).

Таблица 19 - Статистическая значимость различий гибридной модели и DistilBERT

Сравнение	**Δ Accuracy**	**ДИ**	**p-value (McNemar)**
Специализированный корпус	+5,40 п.п.	[+4,8; +6,0]	< 0,001
Синтетический корпус	+0,57 п.п.	[−0,3; +1,4]	0,21
Kaggle Customer Support	0,00 п.п.	[−0,5; +0,5]	1,00

Время обучения гибридной модели составляет от 1,4 до 10 секунд; DistilBERT требует от 600 до 720 секунд - ускорение в 60-514 раз. Число параметров гибридной модели на один-два порядка меньше (1,64-5,0 млн против 66 млн у DistilBERT), что сокращает затраты памяти и вычислений. Гибридная модель работает на CPU и не требует GPU, что снижает стоимость развёртывания и эксплуатации.

Результаты подтверждают применимость гибридной модели в системах модерации контента реального времени, а также в приложениях с ограниченными вычислительными ресурсами.

5.3 Выводы по главе 5

Проведённая в пятой главе экспериментальная оценка эффективности метода квалификации, включающая формирование тестовых данных, сравнение с альтернативными моделями и анализ вклада компонентов, позволила подтвердить практическую реализуемость и воспроизводимость предложенных решений.

1. Показано, что тестовая выборка сформирована из реальных текстовых данных, собранных из открытых интернет-источников в период с января по март 2024 года, общим объёмом 19 855 уникальных записей; распределение по источникам, категориям, сетам, уровням тяжести, тональности и языковому составу обеспечивает репрезентативность.

2. Экспериментальная оценка выполнена на трёх корпусах: специализированном, синтетическом и Kaggle Customer Support. Сравнение с дообученной DistilBERT показало прирост точности на 5,40%. на специализированном корпусе, прирост на 0,57% на синтетическом и сопоставимую точность на наборе данных Kaggle (22,58%).

3. Синергетический эффект достигается за счёт взаимодействия трёх групп методов на разных уровнях: статистические методы извлекают частотные характеристики, нейросетевые предоставляют контекстуализированные векторные представления, лингвистические обеспечивают интерпретируемость.

Таким образом, разработанная библиотека и механизм взаимодействия показывают прирост точности на специализирвоанном наборе на 5,4%. Отличительной особенностью программной реализации является модульная структура подключаемой библиотеки с пятиуровневой обработкой запроса и специализированным механизмом взаимодействия. Полученные результаты подтверждают практическую реализуемость предложенного метода иерархической квалификации и его эффективность при обработке потоковых данных из социальных медиа. В том числе результаты образуют основу для практического применения метода в системах мониторинга информационного пространства и создания интеллектуальных систем обработки потоковых данных.

ЗАКЛЮЧЕНИЕ

В диссертационной работе решена научно-практическая задача повышения точности квалификации неструктурированных текстовых данных за счет разработки гибридного метода, интегрирующего контекстуальные векторные представления, тональный и стилометрический анализ. В процессе исследования получены следующие результаты.

1. Анализ предметной области показал, что существующие подходы к семантическому анализу - символьные, статистические, нейросетевые - обладают взаимодополняющими ограничениями, а пользовательский контент социальных медиа характеризуется лингвистическим шумом, полисемией, быстрой эволюцией языка и субкультурной спецификой. Это определило необходимость перехода к гибридным решениям, удерживающим контекст и выявляющим скрытые семантические связи.

2. Разработана формальная модель квалификации неструктурированных текстовых данных, задаваемая кортежем $M = \langle W, C, R, \Omega, \Phi \rangle$, где:$W$ - множество семантических единиц, C - множество категорий, к которым могут быть отнесены семантические единицы; R - множество отношений между категориями и единицами (иерархические, ассоциативные, логические), Ω - онтологическая спецификация, задающая ограничения на допустимые комбинации категорий и признаков,Φ - функция квалификации, отображающая множество семантических единиц в пространство категорий, где $\Delta(C)$- множество распределений вероятностей на множестве категорий C, отражающее принцип многопринадлежности. Функция может быть реализована в двух режимах: однометочный, используется, когда онтология требует единственной категории и многометочный - каждая категория получает вероятность независимо. Формирование сета выполянется согласно следующими правилу:$Set = f\ (Cat, \Omega)$,где Cat - множество категорий, Ω - онтологическая спецификация. Сет определяется как максимальный класс онтологии, покрывающий все предсказанные категории и не противоречащий аксиомам. Если такого класса нет,

формируется сет «Прочее» с пометкой «Требует подтверждения эксперта». Разработана трёхуровневая архитектура гибридного метода квалификации: первый уровень (признаковый анализ) извлекает количественные характеристики; второй уровень (семантический анализ) вводит контекст, строит семантическую сеть и вычисляет дельту Берроуза как показатель аномальности; третий уровень (интеграция и классификация) принимает итоговое решение на основе свёрточной сети и сопоставляет результат с онтологическими классами.

3. Разработка математического аппарата позволила формализовать все этапы обработки - от BPE-токенизации до формирования сета. BPE обеспечивает эффективное разбиение текста на подсловные единицы и решает проблему редких слов, однако не учитывает семантику, что обосновывает дополнение его алгоритмами семантического и тонального анализа. Предложенное математическое представление алгоритма отличается комплексным объединением контекстуализированных векторных представлений, тонального анализа и модифицированной дельты Берроуза на уровне признакового пространства, а также использованием порядковой регрессии с обучаемыми порогами для раздельной оценки степени тяжести.

4. . Разработана модульная программная структура в виде подключаемой библиотеки Python, рассмотрены преимущества библиотечного подхода. Разработан механизм взаимодействия HQP, обеспечивающий взаимодействие между платформой сбора данных и модулем квалификации и реализующий пять уровней обработки запроса.

5. Проведённая экспериментальная оценка эффективности метода квалификации, включающая формирование тестовых данных, сравнение с альтернативными моделями и анализ вклада компонентов, позволила подтвердить практическую реализуемость и воспроизводимость предложенных решений. Экспериментальная оценка выполнена на трёх корпусах: специализированном, синтетическом и Kaggle Customer Support; сравнение с дообученной DistilBERT показало прирост точности на 5,40%, на специализированном корпусе, прирост на 0,57%,на синтетическом и сопоставимую точность на наборе данных Kaggle

(22,58%). Установлено, что синергетический эффект достигается за счёт взаимодействия трёх групп методов на разных уровнях: статистические методы извлекают частотные характеристики, нейросетевые предоставляют контекстуализированные векторные представления, лингвистические обеспечивают интерпретируемость через онтологическую спецификацию.

Наиболее значимым теоретическим результатом считается формальная модель с иерархией «признак → категория → сет», позволяющая формализовать множественную принадлежность текстов социальных медиа. Разработанный алгоритм комбинирует дельту Берроуза и контекстуальные эмбеддинги для детекции аномалий. Практическая значимость подтверждена внедрением в систему «Деймос»: библиотечная реализация и протокол HQP обеспечивают интеграцию с минимальными накладными расходами, сокращение времени обработки и снижение нагрузки на операторов.

Границы применимости: метод ориентирован на короткие и средние сообщения на русском языке в социальных медиа и мессенджерах. Не исследован на мультимодальном контенте и данных других языковых групп. Дельта Берроуза снижает точность на сообщениях короче 10 слов. Онтологическая модель статична и не предусматривает автоматического обновления. Гиперпараметры подбирались вручную; автоматизированные методы не исследовались. Распараллеливание и кэширование реализованы на уровне одного узла.

Перспективные направления: мультимодальный анализ (визуальные и аудиальные признаки); адаптация к потоковой обработке с горизонтальным масштабированием; визуализация внутренних представлений нейросетей; расширение на другие языки и платформы; автоматизированный подбор гиперпараметров (байесовская оптимизация, обучение с подкреплением).

Таким образом, в работе получены новые научные результаты в области математического и программного обеспечения систем обработки информации. Разработанные модели, методы и алгоритмы создают основу для эффективных систем квалификации неструктурированных текстовых данных, обеспечивающих баланс между точностью, интерпретируемостью, адаптивностью и

вычислительной эффективностью при потоковой обработке данных из социальных медиа. Совокупность результатов показывает, что научно-практическая задача решена: предложенные решения имеют значение для развития методов обработки естественного языка и создания интеллектуальных систем мониторинга информационных потоков в цифровой коммуникационной среде.

СПИСОК СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

Adam - Adaptive Moment Estimation . Адаптивная оценка момента;

API - Application Programming Interface. Интерфейс прикладного программирования;

BERT - Bidirectional Encoder Representations from Transformers. Двунаправленные представления из трансформеров;

BPE - Byte Pair Encoding. Попарное кодирование байтов;

CAPTCHA - Completely Automated Public Turing test to tell Computers and Humans Apart. Полностью автоматизированный публичный тест Тьюринга для различения компьютеров и людей;

CBOW - Continuous Bag-of-Words. Непрерывный мешок слов;

CLIP - Contrastive Language-Image Pre-training. Контрастивное языково-изображенческое предобучение;

CNN - Convolutional Neural Network. Свёрточная нейронная сеть;

DOI - Digital Object Identifier. Цифровой идентификатор объекта;

FAIR - Findable, Accessible, Interoperable, Reusable. Находимость, доступность, совместимость, пригодность для повторного использования;

GDPR - General Data Protection Regulation. Общий регламент по защите данных;

GloVe - Global Vectors for Word Representation. Глобальные векторы для представления слов;

GPT - Generative Pre-trained Transformer. Генеративное предобученное преобразование;

GPU - Graphics Processing Unit. Графический процессор;

HQP - High Qualification Protocol. Протокол квалификации высокого качества;

LDA - Latent Dirichlet Allocation. Латентное размещение Дирихле;

LLM - Large Language Model. Большая языковая модель;

LSTM - Long Short-Term Memory. Долгая краткосрочная память;

NetzDG - Netzwerdgefährdende Gesetze. Закон о сетевой информации (Германия);

ORCID - Open Researcher and Contributor ID. Открытый идентификатор исследователя и автора;

OWL - Web Ontology Language. Язык веб-онтологий;

Python - Язык программирования;

PyTorch - Фреймворк глубокого обучения;

ROR - Research Organization Registry. Реестр исследовательских организаций;

SaaS - Software as a Service. Программное обеспечение как услуга;

scikit-learn - Библиотека машинного обучения для Python;

SVM - Support Vector Machine. Метод опорных векторов;

SWHID - SoftWare Hash IDentifier. Стандарт для создания уникальных, «персистентных» идентификаторов программных артефактов.;

TF-IDF - Term Frequency - Inverse Document Frequency. Частота термина - обратная частота документа;

UML - Unified Modeling Language. Унифицированный язык моделирования;

Word2Vec - Word to Vector. Слово в вектор;

ГБ - Гигабайт;

Гбит/с - Гигабит в секунду;

ГОСТ - Государственный стандарт;

п.п. - Процентный пункт;

РФ - Российская Федерация;

ФЗ - Федеральный закон;

СПИСОК ЛИТЕРАТУРЫ

1. Chomsky N. Syntactic structures. - The Hague : Mouton, 1957. - 117p.

2. Kleene S. C. Representation of events in nerve nets and finite automata // Automata Studies / ed. by C. E. Shannon, J. McCarthy. - Princeton : Princeton University Press, 1956. - P.3-41.

3. Hopcroft J. E., Motwani R., Ullman J. D. Introduction to Automata Theory, Languages, and Computation. - 3rd ed. - Boston : Addison-Wesley, 2007. - 535p.

4. Minsky M. A framework for representing knowledge // MIT-AI Laboratory Memo 306. - Cambridge : MIT, 1974.

5. Harris Z. S. Distributional Structure // Word. - 1954. - Vol. 10, № 2/3. - P. 146-162.

6. Mitchell T. M. Machine Learning. - New York : McGraw-Hill, 1997. - 414 p.

7. Lewis D. D. Naive (Bayes) at forty: The independence assumption in information retrieval // Proceedings of the 10th European Conference on Machine Learning (ECML 98). - Chemnitz : Springer, 1998. - P. 4-15.

8. Joachims T. Text categorization with support vector machines: learning with many relevant features // Proceedings of the 10th European Conference on Machine Learning (ECML 98). - Chemnitz : Springer, 1998. - P. 137-142.

9. Vapnik V. N. The Nature of Statistical Learning Theory. - 2nd ed. - New York : Springer, 2000. - 314 p.

10. Bengio Y., Ducharme R., Vincent P., Janvin C. A neural probabilistic language model // Journal of Machine Learning Research. - 2003. - Vol. 3. - P. 1137-1155.

11. Mikolov T., Chen K., Corrado G., Dean J. Efficient estimation of word representations in vector space // arXiv. - 2013. - arXiv:1301.3781v3.

12. Pennington J., Socher R., Manning C. D. GloVe: Global Vectors for Word Representation // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). - Doha : ACL, 2014. - P. 1532-1543.

13. Levy O., Goldberg Y. Neural word embedding as implicit matrix factorization // Advances in Neural Information Processing Systems 27 (NIPS 2014). - Montreal : Curran Associates, 2014. - P. 2177-2185.

14. Burrows J. F. 'Delta': a measure of stylistic difference and a guide to likely authorship // Literary and Linguistic Computing. - 2002. - Vol. 17, № 3. - P. 267-287.

15. Hoover D. L. Testing Burrows's Delta // Literary and Linguistic Computing. - 2004. - Vol. 19, № 4. - P. 453-475.

16. Eder M. Visualization in stylometry: Cluster analysis using networks // Digital Scholarship in the Humanities. - 2017. - Vol. 32, № 1. - P. 50-64.

17. Kestemont M. Function words in authorship attribution. From black magic to theory? // Proceedings of the 3rd Workshop on Computational Linguistics for Literature. - 2014. - P. 59-66.

18. Sennrich R., Haddow B., Birch A. Neural machine translation of rare words with subword units // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. - 2016. - P. 1715-1725.

19. Provilkov I., Emelianenko D., Voita E. BPE-dropout: Simple and effective subword regularization // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. - 2020. - P. 1882-1892.

20. Stamatatos E. A survey of modern authorship attribution methods // Journal of the American Society for Information Science and Technology. - 2018. - Vol. 60, № 3. - P. 538-556.

21. Neal T., Sundararajan K., Fatima A., Yan Y., Khanna S., Wood S. Surveying stylometry techniques and applications // ACM Computing Surveys. - 2017. - Vol. 50, № 6. - P. 1-36.

22. Schwartz H. A., Eichstaedt J. C., Kern M. L., Dziurzynski L., Ramones S. M., Agrawal M., et al. Personality, gender, and age in the language of social media: The open-vocabulary approach // PLoS ONE. - 2013. - Vol. 8, № 9. - e73791.

23. Sap M., Park G., Eichstaedt J., Kern M., Stillwell D., Kosinski M., et al. Developing age and gender predictive lexica over social media // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. - 2014. - P. 1146-1151.

24. Argamon S., Whitelaw C., Chase P., Hota S. R., Garg N., Levitan S. Stylistic text classification using functional lexical features // Journal of the American Society for Information Science and Technology. - 2007. - Vol. 58, № 6. - P. 802-822.

25. Peng R., Hengartner N. Quantitative analysis of literary styles // The American Statistician. - 2002. - Vol. 56, № 3. - P. 175-185.

26. Kim Y. Convolutional Neural Networks for Sentence Classification // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). - Doha : ACL, 2014. - P. 1746-1751.

27. Johnson R., Zhang T. Supervised and Semi-Supervised Text Categorization using LSTM for Region Embeddings // arXiv. - 2016. - arXiv:1602.02373v1 [cs.LG].

28. Vaswani A., Shazeer N., Parmar N., et al. Attention is all you need // Advances in Neural Information Processing Systems 30 (NIPS 2017). - Long Beach : Curran Associates, 2017. - P. 5998-6008.

29. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // arXiv. - 2018. - arXiv:1810.04805v2 [].

30. Howard J., Ruder S. Universal Language Model Fine-tuning for Text Classification // arXiv. - 2018. - arXiv:1801.06146v5 [].

31. Ruder S. An overview of multi-task learning in deep neural networks // arXiv. - 2017. - arXiv:1706.05098 [cs.LG].

32. Wang W., Hoi S. C. H., Wang J., et al. A comprehensive survey on conversational AI: State of the art, challenges, and future directions // IEEE Transactions on Pattern Analysis and Machine Intelligence. - 2022. - Vol. 44, № 12. - P. 9672-9694.

33. Verberne S., Koster K., Boves L. Text Representations for Patent Classification // Computational Linguistics. - 2024. - URL: https://submissions.cljournal.org/index.php/cljournal/article/view/356 (дата обращения: 01.08.2026).

34. He X., Liu S., Yan G., Zhang X. Text classification method based on dependency parsing and hybrid neural network // Intelligent Data Analysis. - 2024. - P. 1-12. - DOI: 10.3233/IDA-230258.

35. A Study on Patent Literature Classification Using Distributed Representation of Technical Terms // Korean Citation Index. - 2024.

36. Shomee H. H., Wang Z., Ravi S. N., Medya S. A Survey on Patent Analysis: From NLP to Multimodal AI // Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). - Vienna, 2025. - P. 8545-8561. - DOI: 10.18653/v1/2025.acl-long.419.

37. Emer L., et al. Large Language Models for Patent Classification: Strengths, Trade-offs, and the Long Tail Effect // arXiv. - 2026. - arXiv:2601.23200 [cs.CE].

38. Erhardt S. Automated Patent Landscaping Using Deep Learning // Proceedings of the 5th Workshop on Patent Text Mining and Semantic Technologies (PatentSemTech 2024). - Washington, D.C., 2024.

39. Xiong S., Chen S., He J., Liu Y., Mao J., Liu C. Scalable multi-label patent classification via iterative large language model-assisted active learning // World Patent Information. - 2025. - DOI: 10.1016/j.wpi.2025.102388.

40. Сяо Дун С., У Цзе, Шэн Юнсян, Тан Цзяньтин, Юй Юй. Research on Automatic Patent Classification Method Based on Multimodal Deep Learning // Journal of Intelligence. - 2025. - Vol. 44, No. 5. - P. 199-206

41. Shomee H. H., et al. DesignCLIP: A Unified Framework for Design Patent Analysis // Findings of the Association for Computational Linguistics: EMNLP 2025. - 2025.

42. Choi S., Lee H., Park E., Choi S. Deep Patent Landscaping Model Using the Transformer and Graph Embedding // arXiv. - 2019. - arXiv:1903.05823 []. - URL: https://arxiv.org/abs/1903.05823 (дата обращения: 01.08.2026).

43. Zang W., Ma W., Chen Y., Zhao Y., Wang Y., Liu X., Chen Y. TextCG: A text classification framework based on information fusion via cross-graph attention network and gated recurrent unit // Information Sciences. - 2025. - Vol. 719. - P. 122413. - DOI: 10.1016/j.ins.2025.122413.

44. Kuzman T., Ljubešić N. LLM Teacher-Student Framework for Text Classification With No Manually Annotated Data: A Case Study in IPTC News Topic

Classification // IEEE Access. - 2024. - Vol. 12. - P. 187634-187652. - DOI: 10.1109/ACCESS.2024.3508727.

45. Wang Y., Li J., Zhang H., Liu S., Chen W. A survey on text classification in corporate information systems: from machine learning to large language models // Journal of Industrial Information Integration. - 2024. - Vol. 42. - P. 100712. - DOI: 10.1016/j.jii.2024.100712.

46. Wu Y., Wan J. A survey of text classification based on pre-trained language model // Neurocomputing. - 2025. - Vol. 616. - Art. 128921. - DOI: 10.1016/j.neucom.2024.128921.

47. Weaver W. Translation // Memorandum. - New York : Rockefeller Foundation, 1949. - 12 p.

48. Jurafsky D., Martin J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. - 3rd ed. draft. - [Б. м.] : авт. изд., 2021. - URL: https://web.stanford.edu/~jurafsky/slp3/ (дата обращения: 01.08.2026).

49. Manning C. D., Raghavan P., Schütze H. Introduction to Information Retrieval. - Cambridge : Cambridge University Press, 2008. - 482 p.

50. Aggarwal C. C., Zhai C. X. Mining Text Data. - New York : Springer, 2012. - 522 p.

51. Haykin S. Neural Networks: A Comprehensive Foundation. - 2nd ed. - Upper Saddle River : Prentice Hall, 1999. - 842 p.

52. Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain // Psychological Review. - 1958. - Vol. 65, № 6. - P. 386-408.

53. Hebb D. O. The Organization of Behavior: A Neuropsychological Theory. - New York : Wiley & Sons, 1949. - 335 p.

54. Zhang Z., Sabuncu M. Peer Loss: Learning from Noisy Labels without Knowing Noise Rates // Proceedings of the 37th International Conference on Machine Learning (ICML 2020). - Virtual : PMLR, 2020. - P. 11109-11119.

55. Rizvi S. M. H., Imran R., Mahmood A. Text Classification Using Graph Convolutional Networks: A Comprehensive Survey // ACM Computing Surveys. - 2025. - Vol. 57, № 8. - Art. 201. - DOI: 10.1145/3714456.

56. Sakai H., Lam S. S. Large Language Models for Healthcare Text Classification: A Systematic Review // arXiv. - 2025. - arXiv:2503.01159 [].

57. Taha K., Yoo P. D., Yeun C., Taha A. Text Classification Techniques: A Holistic Review, Observational Analysis, and Experimental Investigation // Big Data Mining and Analytics. - 2025. - Vol. 8, № 3. - P. 624-660. - DOI: 10.26599/BDMA.2024.9020081.

58. Kiros R., Salakhutdinov R., Zemel R. S. Unifying visual-semantic embeddings with multimodal neural language models // arXiv. - 2014. - arXiv:1411.2539v1 [cs.LG].

59. Socher R., Karpathy A., Le Q. V., et al. Grounded compositional semantics for finding and describing images with sentences // Transactions of the Association for Computational Linguistics. - 2014. - Vol. 2. - P. 207-218.

60. Collobert R., Weston J. A unified architecture for natural language processing: deep neural networks with multitask learning // Proceedings of the 25th International Conference on Machine Learning (ICML '08). - Helsinki : ACM, 2008. - P. 160-167.

61. Liu B. Sentiment Analysis: Mining Opinions, Sentiments, and Emotions. - 2nd ed. - Cambridge : Cambridge University Press, 2020. - 448 p.

62. Girdhar R., El-Nouby A., Liu Z., et al. ImageBind: One embedding space to bind them all // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). - 2023. - P. 15180-15190.

63. Bojanowski P., Grave E., Joulin A., Mikolov T. Enriching word vectors with subword information // Transactions of the Association for Computational Linguistics. - 2017. - Vol. 5. - P. 135-146. - DOI: 10.1162/tacl_a_00051.

64. Ribeiro M. T., Singh S., Guestrin C. «Why Should I Trust You?»: Explaining the predictions of any classifier // Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). - San Francisco : ACM, 2016. - P. 1135-1144.

65. Selvaraju R. R., Cogswell M., Das A., et al. Grad-CAM: Visual explanations from deep networks via gradient-based localization // Proceedings of the IEEE International Conference on Computer Vision (ICCV). - 2017. - P. 618-626.

66. Rudin C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead // Nature Machine Intelligence. - 2019. - Vol. 1, № 5. - P. 206-215.

67. Kairouz P., McMahan H. B., Avent B., et al. Advances and open problems in federated learning // arXiv. - 2019. - arXiv:1912.04977 [cs.LG].

68. Тотмина Е. В., Бондаренко И., Серёдкин А. В. Методология иерархического многозадачного обучения нейронных сетей типа ERNIE 3 для анализа и генерации русскоязычных текстов // Моделирование и анализ информационных систем. - 2025. - Т. 32, № 3. - С. 282-297. - DOI: 10.18255/1818-1015-2025-3-282-297.

69. Zhang X., Zhao J., LeCun Y. Character-level convolutional networks for text classification // Advances in Neural Information Processing Systems 28 (NIPS 2015). - 2015. - URL: https://arxiv.org/abs/1509.01626 (дата обращения: 01.08.2026).

70. Kowsari K., Meimandi K. J., Heidarysafa M., et al. Text classification algorithms: a survey // Information. - 2019. - Vol. 10, № 4. - Art. 150. - DOI: 10.3390/info10040150.

71. Zhang Y., Wallace B. C. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification // Proceedings of the Eighth International Joint Conference on Natural Language Processing (IJCNLP 2017). - Taipei : AFNLP, 2017. - P. 253-263.

72. Minaee S., Kalchbrenner N., Cambria E., et al. Deep learning based text classification: a comprehensive review // arXiv. - 2020. - arXiv:2004.03705 [].

73. Mikolov T., Sutskever I., Chen K., et al. Distributed representations of words and phrases and their compositionality // Advances in Neural Information Processing Systems 26 (NIPS 2013). - Lake Tahoe : Curran Associates, 2013. - P. 3111-3119.

74. Смирнов И. В. Интеллектуальный анализ текстов на основе методов разноуровневой обработки естественного языка : монография. - Москва : ФИЦ ИУ РАН, 2023. - 354 с. - ISBN 978-5-6050647-0-1.

75. Соченков И. В. Методы и алгоритмы поиска и анализа текстовой информации : монография. - Москва : ТОРУС ПРЕСС, 2018. - 216 с. - ISBN 978-5-94588-236-9.

76. Маннинг К., Рагхаван П., Шютце Х. Введение в информационный поиск. - Москва : Вильямс, 2011. - 528 с. - ISBN 978-5-8459-1623-5.

77. Агеев М. С., Добров Б. В., Лукашевич Н. В. Методы автоматической рубрикации текстов : монография. - Москва : Издательство Московского университета, 2014. - 184 с. - ISBN 978-5-19-010877-4.

78. Лагутина К. В., Бойчук Е. И., Лагутина Н. С. Автоматическая классификация русскоязычных интернет-текстов по жанрам // Искусственный интеллект и принятие решений. - 2023. - № 4. - С. 103-114. - DOI: 10.14357/20718594230410.

79. Логинова И. В., Пиекалнитс А. С., Сабидаева Е. А., и др. Содержательные критерии отнесения утверждений из текстов к категориям «события» и «факторы» // Искусственный интеллект и принятие решений. - 2024. - № 4. - С. 93-110. - DOI: 10.14357/20718594240408.

80. Жданова А. Н., Куприянов А. В., Голова А. А., и др. Исследование эффективности различных архитектур рекуррентных нейронных сетей для решения задачи анализа тональности русскоязычных комментариев пользователей социальных сетей // Автометрия. - 2023. - Т. 59, № 4. - С. 29-38. - DOI: 10.15372/AUT20230404.

81. Челышев Э. А., Оцоков Ш. А., Раскатова М. В., Щёголев П. Сравнение методов классификации русскоязычных новостных текстов с использованием алгоритмов машинного обучения // Вестник кибернетики. - 2022. - № 1 (45). - С. 22-29.

82. Васильев А. А., Нестеров А. С. Применение алгоритмов формирования вопросов по тексту для автоматической генерации тестов // Вестник кибернетики. - 2023. - Т. 22, № 3. - С. 6-13.

83. Кожемякина О. Ю., Барахнин В. Б. Объединенный семинар «Информационно-вычислительные технологии в задачах филологии и компьютерной лингвистики». Аннотации докладов за 2023 г. // Вычислительные технологии. - 2024. - Т. 29, № 2. - С. 113-120.

84. Бондаренко А. В., Котельников Е. В. Сравнительный анализ методов векторизации текстов для задачи классификации // Программная инженерия. - 2021. - Т. 12, № 5. - С. 237-245. - DOI: 10.17587/prin.12.237-245.

85. Collobert R., et al. Natural Language Processing (Almost) from Scratch // Journal of Machine Learning Research. - 2011. - Vol. 12. - P. 2493-2537.

86. Blei D. M., Ng A. Y., Jordan M. I. Latent Dirichlet Allocation // Journal of Machine Learning Research. - 2003. - Vol. 3. - P. 993-1022.

87. Mohammad S. M., Turney P. D. Crowdsourcing a Word-Emotion Association Lexicon // Computational Intelligence. - 2013. - Vol. 29, № 3. - P. 436-465.

88. Bridle J. S. Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition // Neurocomputing. - Springer, 1990. - P. 227-236.

89. Caruana R. Multitask Learning // Machine Learning. - 1997. - Vol. 28, № 1. - P. 41-75.

90. Verhulst P. F. Notice sur la loi que la population suit dans son accroissement // Correspondance Mathématique et Physique. - 1838. - Vol. 10. - P. 113-121.

91. Ghosh A., Veale T. Fracking Sarcasm Using Neural Network // Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA 2016). - 2016. - P. 161-169.

92. Kumar A., Narapareddy V. T., et al. Sarcasm Detection Using Multi-Head Attention and Multi-Task Learning // IEEE Transactions on Affective Computing. - 2020. - Vol. 11, № 4. - P. 1-12.

93. Filatova E. Irony and Sarcasm: Corpus Generation and Analysis // Literary and Linguistic Computing. - 2012. - Vol. 27, № 1. - P. 23-36.

94. Wallace B. C., et al. Humor, Sarcasm, and Irony in Text // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). - 2014. - P. 176-186.

95. Bamman D., Smith N. A. Contextualized Sarcasm Detection on Twitter // Proceedings of the Ninth International AAAI Conference on Web and Social Media (ICWSM). - 2015. - P. 35-43.

96. Клышинский Э. С., Кочеткова Н. А., Мансурова О. Ю. Обзор методов автоматического реферирования текстов // Новые информационные технологии в автоматизированных системах. - 2020. - № 23. - С. 177-186.

97. Лукашевич Н. В., Морозова Ю. И. Извлечение терминов и связей между ними из научно-технических текстов // Искусственный интеллект и принятие решений. - 2021. - № 2. - С. 45-57. - DOI: 10.14357/20718594210205.

98. Соловьев В. Д., Бочкарев В. В., Шевлякова А. В. Семантический анализ текстов с использованием дистрибутивной семантики и нейронных сетей // Ученые записки Казанского университета. Серия: Физико-математические науки. - 2019. - Т. 161, № 1. - С. 131-144.

99. Захарова А. В., Вишнякова А. Ю., Детков А. А. Сравнение моделей и методов классификации текста // Вестник РУДН. Серия: Инженерные исследования. - 2025. - Т. 26, № 3. - С. 298-309. - DOI: 10.22363/2312-8143-2025-26-3-298-309.

100. Pham P., Nguyen L. T. T., Pedrycz W., Vo B. Deep learning, graph-based text representation and classification: a survey, perspectives and challenges // Artificial Intelligence Review. - 2023. - Vol. 56, № 6. - P. 4893-4927. - DOI: 10.1007/s10462-023-10549-6.

101. Malykh V., Filchenkov A., eds. Artificial Intelligence and Natural Language: 11th Conference, AINL 2022, Saint Petersburg, Russia, April 14-15, 2022: revised selected papers. - Cham : Springer, 2023. - 109 p. - (Communications in Computer and Information Science). - ISBN 978-3-031-23854-8.

102. Иномов Б. Б., Тропманн-Фрик М. КЛАССИФИКАЦИЯ НАУЧНЫХ ТЕКСТОВ ПО СПЕЦИАЛЬНОСТЯМ МЕТОДАМИ МАШИННОГО ОБУЧЕНИЯ // Вестник НГУ. Серия: Информационные технологии. 2022. №2. URL: https://cyberleninka.ru/article/n/klassifikatsiya-nauchnyh-tekstov-po-spetsialnostyam-metodami-mashinnogo-obucheniya (дата обращения: 01.08.2026).

103. Тутубалина Е.В., Иванов В.В., Загулова М., Мингазов Н., Алимова И., Малых В. Тестирование методов анализа тональности текста, основанных на словарях. Электронные библиотеки. 2015;18(3-4):138-162. - URL: https://ellibs.elpub.ru/jour/article/view/433 (дата обращения: 01.08.2026).

104. Копылов Д. Е., Михайлов А. А., Трифонов Р. И. Сегментация документов на основе графовых нейронных сетей: от строк к словам // Труды Института системного программирования РАН. - 2025. - Т. 37, № 6(1). - С. 219-232. - DOI: 10.15514/ISPRAS-2025-37(6)-14.

105. Хлебникова А. А., Битюцкая Е. В., Калачев Г. В., Гасанов Э. Э. Автоматизация разметки текстов о жизненных трудностях с использованием больших языковых моделей // Интеллектуальные системы. Теория и приложения. - 2025. - Т. 29, № 3. - С. 53-75.

106. Kostina A., Dikaiakos M. D., Stefanidis D., Pallis G. Large Language Models For Text Classification: Case Study And Comprehensive Review // arXiv. - 2025. - arXiv:2501.08457.

107. Sarin G., Kumar P., Mukund M. Text classification using deep learning techniques: a bibliometric analysis and future research directions // Benchmarking: An International Journal. - 2023. - Vol. 30, № 8. - P. 2743-2766. - DOI: 10.1108/BIJ-09-2021-0526.

108. Liu J., Hu S., Mehraliyev F., Liu H. Text classification in tourism and hospitality - a deep learning perspective // International Journal of Contemporary Hospitality Management. - 2023. - DOI: 10.1108/IJCHM-02-2023-0191.

109. Котов Э. М. Автоматическая классификация текстовых документов // Известия ЮФУ. Технические науки. - 2005. - № 9. - URL:

https://cyberleninka.ru/article/n/avtomaticheskaya-klassifikatsiya-tekstovyh-dokumentov (дата обращения: 01.08.2026).

110. Acikgoz E. C., Qian C., Wang H., et al. A Desideratum for Conversational Agents: Capabilities, Challenges, and Future Directions // arXiv. - 2025. - arXiv:2504.16939 [cs.HC].

111. Crawshaw M. Multi-Task Learning with Deep Neural Networks: A Survey // arXiv. - 2020. - arXiv:2009.09796 [cs.LG].

112. Wang Y., Nurbakova D., Calabretto S. Team INSAntive at SemEval-2025 Task 10: Hierarchical Text Classification using BERT // Proceedings of the 19th International Workshop on Semantic Evaluation (SemEval 2025). - URL: https://hal.science/hal-05154004v1/document (дата обращения: 01.08.2026).

113. Коренева А. В., Рыжкова И. В., Гурылева А. В., Сергеева Л. А. Анализ текста: лингвистические и методические аспекты : монография. - Москва : РУСАЙНС, 2024. - 192 с. - ISBN 978-5-466-07472-7.

114. Brandwatch : [сайт]. - [Б. м.], 2007-2026. - URL: https://www.brandwatch.com/ (дата обращения: 01.08.2026).

115. Brand Analytics : [сайт]. - Москва, 2013-2026. - URL: https://brandanalytics.ru/ (дата обращения: 01.08.2026).

116. Flanagan D. JavaScript: The Definitive Guide. - 7th ed. - Sebastopol : O'Reilly Media, 2020. - 704 p.

117. Hejlsberg A., Torgersen M., Wiltamuth S., Golde P. The C# Programming Language. - 4th ed. - Boston : Pearson, 2010. - 848 p.

118. Lutz M. Learning Python. - 5th ed. - Sebastopol : O'Reilly Media, 2013. - 1648 p.

119. Ecma International. ECMAScript® 2025 Language Specification (ECMA-262, 14th edition). - Geneva : Ecma International, 2025. - URL: https://www.ecma-international.org/publications-and-standards/standards/ecma-262/ (дата обращения: 01.08.2026).

120. ISO/IEC 14882:2020. Programming languages - Geneva : ISO, 2020. - 1853 p.

121. Российская Федерация. Законы. О персональных данных : Федеральный закон № 152-ФЗ : [принят Гос. Думой 8 июля 2006 г. : одобрен Советом Федерации 14 июля 2006 г.] // Собрание законодательства Российской Федерации. - 2006. - № 31 (ч. 1). - Ст. 3451.

122. Российская Федерация. Законы. Об информации, информационных технологиях и о защите информации: Федеральный закон № 149-ФЗ : [принят Гос. Думой 8 июля 2006 г. : одобрен Советом Федерации 14 июля 2006 г.] // Собрание законодательства Российской Федерации. - 2006. - № 31 (ч. 1). - Ст. 3448.

123. Российская Федерация. Законы. О связи: Федеральный закон № 126-ФЗ : [принят Гос. Думой 18 июня 2003 г. : одобрен Советом Федерации 25 июня 2003 г.] // Собрание законодательства Российской Федерации. - 2003. - № 28. - Ст. 2895.

124. Российская Федерация. Законы. О защите детей от информации, причиняющей вред их здоровью и развитию : Федеральный закон № 436-ФЗ : [принят Гос. Думой 21 дек. 2010 г. : одобрен Советом Федерации 24 дек. 2010 г.] // Российская газета. - 2010. - 29 дек. (№ 5376).

125. Российская Федерация. Законы. О внесении изменений в Федеральный закон «О связи» : Федеральный закон № 90-ФЗ : [принят Гос. Думой 16 апр. 2019 г. : одобрен Советом Федерации 22 апр. 2019 г.] // Официальный интернет-портал правовой информации. - URL: http://publication.pravo.gov.ru/Document/View/0001201905010041 (дата обращения: 01.08.2026).

126. Соединенные Штаты Америки. Законы. Communications Decency Act of 1996 (47 U.S.C. § 230) : [принят Конгрессом США 1 февр. 1996 г.] // United States Code. - Washington : U.S. Government Publishing Office, 1996 (с изм. на 2025 г.). - URL: https://www.law.cornell.edu/uscode/text/47/230 (дата обращения: 01.08.2026).

127. Европейский Союз. Регламент Европейского Парламента и Совета Европейского Союза 2016/679 от 27 апреля 2016 г. о защите физических лиц при обработке персональных данных и о свободном обращении таких данных, а также об отмене Директивы 95/46/ЕС (Общий регламент о защите данных) // Официальный журнал Европейского Союза. - 2016. - L 119. - С. 1-88.

128. Китайская Народная Республика. Законы. Закон Китайской Народной Республики о кибербезопасности : [принят на 24-й сессии Постоянного комитета ВСНП 7 нояб. 2016 г. : вступил в силу 1 июня 2017 г.]// Официальный сайт Правительства КНР. - URL: http://www.gov.cn/xinwen/2016-11/07/content_5129723.htm (дата обращения: 01.08.2026).

129. Германия. Законы. Закон о повышении правоприменения в социальных сетях (Netzwerkdurchsetzungsgesetz - NetzDG) : от 1 сент. 2017 г. (BGBl. I S. 3350) : в ред. от 28.06.2021 (BGBl. I S. 2258) // Bundesgesetzblatt. - 2017. - Teil I, № 61. - С. 3350-3355.

130. McNemar Q. Note on the sampling error of the difference between correlated proportions or percentages // Psychometrika. - 1947. - Vol. 12, № 2. - P. 153-157.

131. Customer Support Ticket Dataset (Kaggle https://www.kaggle.com/datasets/suraj520/customer-support-ticket-dataset (дата обращения: 01.08.2026).

132. Алексеева Д. С., Кононов Н. А. Криптография средствами офисного пакета MS Excel: возможности и оценка реализации // World science: problems and innovations : сборник статей LXVII Международной научно-практической конференции. - Пенза, 2022. - С. 36-39.

133. Алексеева Д. С. Обзор юридического сопровождения средств криптографической защиты // Мавлютовские чтения : материалы XVI Всероссийской молодежной научной конференции. В 6-ти томах. - Уфа, 2022. - С. 451-455.

134. Алексеева Д. С., Кононов Н. А. Разработка и оценка реализации алгоритма шифрования RSA с применением языка Visual Basic for Applications // Молодежный вестник УГАТУ. - 2023. - № 1 (27). - С. 10-14.

135. Алексеева Д. С., Антонов В. В., Харисова З. И., Ишмеева А. С. Возможности применения средств видеофиксации с обработкой данных на основе искусственного интеллекта при производстве следственных действий // IV Международная научная конференция по междисциплинарным исследованиям : сборник статей. - Екатеринбург, 2023. - С. 374-380.

136. Алексеева Д. С., Кононов Н. А. Подход к автоматизации процесса выбора программного средства защиты информации // Молодежный вестник УГАТУ. - 2024. - № 1 (30). - С. 5-9.

137. Алексеева Д. С., Суворова В. А. Метод обеспечения защиты системы автоматизированного сбора данных из открытых источников от встречного воздействия исследуемого объекта // Информационные технологии интеллектуальной поддержки принятия решений (памяти проф. Н. И. Юсуповой) ITIDS'2024 : труды X Международной научной конференции. В 2-х томах. - Уфа, 2024. - С. 284-289.

138. Алексеева Д. С., Нуриахметов А. И., Юлина Е. В. Интеллектуальные технологии автоматизированного сбора и анализа данных в технических приложениях: архитектура, методы, перспективы // Лучшие научные исследования студентов и учащихся : сборник статей XI Международной научно-практической конференции. - Пенза, 2025. - С. 78-83.

139. Алексеева Д. С., Кононов Н. А., Антонов В. В., Еникеев Р. Р. Алгоритм анализа сетевого трафика в многокомпонентных структурно сложных информационных системах // Вестник УГАТУ. - 2024. - Т. 28, № 1(103). - С. 92-98.

140. Алексеева Д. С., Пальчевский Е. В., Антонов В. В., Суворова В. А. Оптимизация математической основы IRT с помощью LLM-моделей // Системная инженерия и информационные технологии. - 2026. - Т. 8, № 1(25). - С. 03-18.

141. Алексеева Д. С. Мультипараметрический метод установления авторства текста: интеграция стилометрического, тематического и прагматического анализа // Системная инженерия и информационные технологии. - 2026. - Т. 8, № 1(25). - С. 75-83.

142. Алексеева Д. С., Юлина Е. В., Кононов Н. А., Антонов В. В., Дьячков А. С. Разработка модуля автоматизированной предобработки текстовых данных с адаптивным управлением для нейросетевых моделей // Вестник УГАТУ. - 2026. - Т. 30, № 2. - С. 41-48.

143. Алексеева Д. С. Свидетельство о регистрации программы для ЭВМ № RU 2023661529 «Программа автоматизации деятельности торгового агента» // Зарег. 01.06.2023. - М. : Роспатент, 2023.

144. Алексеева Д. С. Свидетельство о регистрации программы для ЭВМ № RU 2023661948 «Имитационная модель средства ирригации» // Зарег. 05.06.2023. - М. : Роспатент, 2023.

145. Нуриахметов А. И., Алексеева Д. С., Еникеев Р. Р. Свидетельство о регистрации базы данных № RU 2025623973 «База данных автоматизированного тестирования программных модулей» // Зарег. 23.09.2025. - М. : Роспатент, 2025.

146. Мешкова Е. В. Методика построения классификатора текста на основе гибридной нейросетевой модели // Известия ЮФУ. Технические науки. 2008. №4. URL: https://cyberleninka.ru/article/n/metodika-postroeniya-klassifikatora-teksta-na-osnove-gibridnoy-neyrosetevoy-modeli (дата обращения: 01.08.2026)

147. Sangmin Bae, Bilge Acun, Haroun Habeeb, Seungyeon Kim, Chien-Yu Lin, Liang Luo, Junjie Wang, Carole-Jean Wu Гибридные архитектуры для языковых моделей: систематический анализ и принципы проектирования // - URL: https://arxiv.org/pdf/2510.04800 (дата обращения: 01.08.2026)

ПРИЛОЖЕНИЕ А

Акты о внедрении научных результатов

Акт о внедрении в ООО «Центр Информационных технологий»

ООО "Центр информационных технологий"
ИНН 0277128392 КПП 027601001
Адрес: г. Уфа, ул. Новоженова, д.88

УТВЕРЖДАЮ
директор
ООО "Центр информационных технологий"
_____/С.О. Скалдин
«09» 2026г.

АКТ

о внедрении результатов диссертационной работы Алексеевой Д.С. «Модель, алгоритм и программное средство квалификации неструктурированных данных в автоматизированных системах сбора информации»

Экспертная комиссия в составе: директор ООО «Центр информационных технологий» Скалдин Станислав Олегович, директор по разработке ООО «Центр информационных технологий» Титов Сергей Викторович, заместитель директора Ахметзянов Марат Салаватович составили настоящий акт о том, что результаты диссертационной работы «Модель, алгоритм и программное средство квалификации неструктурированных данных в автоматизированных системах сбора информации» Алексеевой Дарьи Сергеевны, представленной на соискание ученой степени кандидата технических наук, внедрены в промышленную информационно-аналитическую систему «Деймос».

Результаты диссертационной работы реализованы в виде библиотеки языка Python. Внедрение затронуло основные процессы обработки контента социальных медиа, получаемых из открытых источников информации.

В основу процесса внедрения положены разработанные автором:

– формальная модель квалификации неструктурированных текстовых данных с иерархией «признак → категория → сет» и механизмом множественной принадлежности семантических единиц;

– алгоритм гибридной квалификации, объединяющий контекстуализированные векторные представления, тональный анализ, стилометрическую оценку аномальности (модифицированная дельта Берроуза), детекцию сарказма и порядковую регрессию для оценки степени тяжести;

– программную реализацию в виде подключаемой библиотеки Python, включающей ядро гибридной модели;

– механизм онтологической валидации результатов классификации с проверкой аксиом и правил онтологии.

Применение модели и алгоритма гибридной квалификации позволило добиться значительных улучшений в работе подразделения, осуществляющего мониторинг и анализ информационного пространства: сокращено время на анализ и принятие решений по поступающим сообщениям за счёт автоматической категоризации и оценки степени тяжести (точность категоризации контента повышена на 5,4%, точность оценки степени тяжести контента), доля объектов, требующих ручной проверки, сокращена до 0,8% за счёт онтологической валидации контента; время на согласование и актуализацию ежедневной отчетности за счёт прямой связи между онтологическими моделями и регламентами сокращено

на 20%; повысилась согласованность действий между аналитическими, техническими и профильными подразделениями, что снизило количество спорных ситуаций и пересмотров решений. Обеспечена прозрачность и прослеживаемость всех решений по квалификации контента, что сократило время на подготовку доказательной базы.

Таким образом, внедрение результатов диссертационной работы «Модель, алгоритм и программное средство квалификации неструктурированных данных в автоматизированных системах сбора информации», позволило создать интеллектуальную, объяснимую и адаптивную систему квалификации неструктурированных текстовых данных, интегрированную в процессы обработки информации системой «Деймос».

Председатель комиссии Директор		/С.О. Скалдин/
Директор по разработке		/С.В. Титов/
Заместитель директора		/М.С. Ахметзянов/
Аналитик		/А.А. Овчинникова/

Акт о внедрении в АО «Центр информационно-коммуникационных технологий Республики Башкортостан»

АКТ

о внедрении результатов диссертационной работы Алексеевой Д.С. «Модель, алгоритм и программное средство квалификации неструктурированных данных в автоматизированных системах сбора информации»

Экспертная комиссия в составе:

Председатель комиссии:

– Генеральный директор АО «Центр ИКТ РБ» Скатов Алексей Константинович;

Члены комиссии:

– Заместитель генерального директора АО «Центр ИКТ РБ» по информационным технологиям Ключеров Роман Викторович;

– Начальник отдела электронного документооборота Биккинин Ильнур Фагимович составили настоящий акт о том, что результаты диссертационной работы «Модель, алгоритм и программное средство квалификации неструктурированных данных в автоматизированных системах сбора информации» Алексеевой Дарьи Сергеевны, представленной на соискание ученой степени кандидата технических наук, внедрены в работу АО «Центр информационно-коммуникационных технологий Республики Башкортостан».

Научные положения диссертации были применены при создании модуля интеллектуальной квалификации неструктурированных текстовых данных для систем мониторинга информационного пространства, а также разработаны, протестированы и внедрены в сфере автоматизированного анализа пользовательского контента, программной обработки текстовых потоков и интеграции гетерогенных данных.

Центральным элементом внедрения стала библиотека гибридной квалификации текстовых данных, интегрированная с корпоративными системами сбора и обработки информации, что позволило формализовать экспертные знания и сократить сроки разработки сценариев анализа. Кроме того, были использованы следующие элементы из диссертации: формальная модель квалификации с иерархией «признак → категория → сет», алгоритм гибридной квалификации, объединяющий контекстуализированные векторные представления, тональный анализ и модифицированная дельта Берроуза.

В результате внедрения достигнуты улучшения в процессах обработки текстовых данных: время на категоризацию и оценку степени тяжести сообщений сократилось на 5-10%; время на подготовку объяснений и проверку решений уменьшилось на 15-20%

Научные положения диссертации доказали свою практическую применимость и обеспечили повышение эффективности в задачах формализации знаний, поддержки принятия решений, автоматизированной обработки текстов и онтологического моделирования.

Настоящим актом подтверждается, что результаты диссертационного исследования на тему «Модель, алгоритм и программное средство квалификации неструктурированных данных в автоматизированных системах сбора информации» внедрены в АО «Центр информационно-коммуникационных технологий Республики Башкортостан».

Председатель комиссии
Генеральный директор
АО «Центр ИКТ РБ» _____ /А.К. Скатов /

Члены комиссии:
Заместитель генерального директора по информационным технологиям
АО «Центр ИКТ РБ» _____ / Ключеров Р.В./

Начальник отдела электронного документооборота _____ / Биккинин И.Ф. /

22.09.2026г.

Акт об использовании в учебном процессе ФГБО ВО «Уфимский университет науки и технологий»

УТВЕРЖДАЮ
Проректор по развитию образования
ФГБОУ ВО Уфимский университет
науки и технологий Рахманова Ю.В.
«21» 09 2026г.

АКТ
ОБ ИСПОЛЬЗОВАНИИ

результатов диссертационной работы Алексеевой Дарьи Сергеевны на тему «**Модель, алгоритм и программное средство квалификации неструктурированных данных в автоматизированных системах сбора информации**», представленной на соискание ученой степени кандидата технических наук

Мы, нижеподписавшиеся, директор института информатики математики и робототехники Кривошеева О.А. и заведующий кафедрой автоматизированных систем управления Антонов В.В. составили настоящий акт о том, что основные результаты диссертационной работы Алексеевой Д.С. (алгоритм гибридной квалификации текстового контента, основанный на совместном применении контекстуализированных векторных представлений, тонального анализа и стилометрической метрики) внедрены в учебный процесс кафедры автоматизированных систем управления и используются в учебном процессе по дисциплинам специалитета «Применение и эксплуатация автоматизированных систем специального назначения» (09.05.01): «Специальные информационные аналитические системы» (Анализ потоковых данных в реальном времени), «Автоматизированные системы специального назначения» (Моделирование системы сбора данных в SCADA-системе).

В результате подтверждена практическая применимость алгоритма гибридной квалификации и применимость программной реализации подключаемой библиотеки Python, включающей ядро гибридной модели. Студенты освоили современные подходы к семантическому анализу и решению задач квалификации текстов в алгоритмах анализа потоковых данных.

Заведующий кафедрой АСУ
д.т.н.
/В.В. Антонов/
Дата: «21» 09 2026г

Директор института информатики, математики и робототехники, д.ф-м.н
/О.А. Кривошеева/
Дата: «21» 09 2026г.

ПРИЛОЖЕНИЕ Б

Блок - схема предлагаемого решения

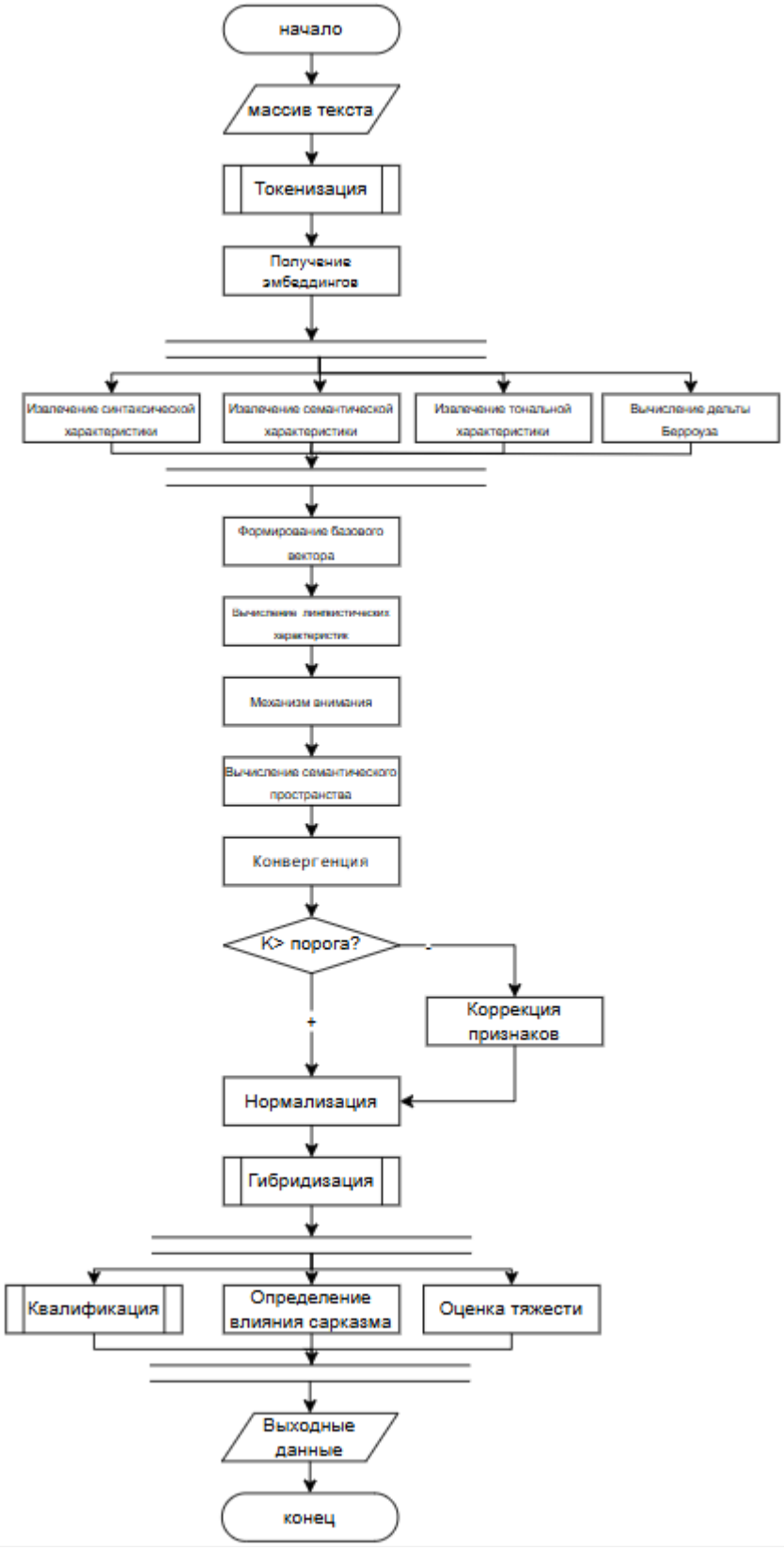


Таблица Б1 - Соответствие формул и блоков алгоритма

Формула	**Содержание**	**Блок на схеме**
(1)	BPE-токенизация исходного текста с построением последовательности целочисленных идентификаторов токенов	Процесс «Токенизация»
(2)	Построение матрицы эмбеддингов токенов размерности $n \times d$	Процесс «Получение эмбеддингов»
(3)	Извлечение синтаксических признаков с использованием одномерной сверточной нейронной сети	Процесс «Извлечение синтаксической характеристики»
(4)	Извлечение семантических признаков методом латентного размещения Дирихле	Процесс «Извлечение семантической характеристики»
(5)-(6)	Вычисление лексической тональности и итоговой трехкомпонентной оценки тональности	Процесс «Извлечение тональной характеристики»
(7)-(8)	Вычисление дельты Берроуза и нормирование показателя аномальности в диапазон [0, 1]	Процесс «Вычисление дельты Берроуза»
(9)	Вычисление косинусного расстояния между начальной и конечной частями текста	Процесс «Вычисление лингвистических характеристик»
(10)-(13)	Извлечение именных групп, вычисление тональности темы и контекста, определение индекса семантической инконгруэнтности	Процесс «Вычисление лингвистических характеристик»
(14)	Вычисление коэффициента эмоциональной интенсивности на основе контраста эмоциональных оценок соседних токенов	Процесс «Вычисление лингвистических характеристик»
(15)	Вычисление индекса гиперболизации на основе словаря гипербол	Процесс «Вычисление лингвистических характеристик»
(16)	Вычисление коэффициента стилистического диссонанса относительно эталонного стиля темы	Процесс «Вычисление лингвистических характеристик»
(17)-(19)	Вычисление весов внимания, вектора контекстной противоречивости и скаляра саркастичности	Процесс «Механизм внимания»
(20)	Вычисление коэффициента контекстного диссонанса для диалоговых сообщений	Процесс «Механизм внимания»
(21)-(23)	Построение семантического пространства текста, вычисление коэффициента конвергенции и нормализация семантического пространства	Процесс «Вычисление семантического пространства» и «Конвергенция»
(24)-(25)	Формирование расширенного интегрального вектора с регуляризацией аномальностью и Z-стандартизация	Процесс «Формирование интегрального вектора» и «Нормализация»
(26)	Нейросетевая оценка категорий с использованием линейного классификатора и softmax-нормализации	Процесс «Гибридизация»
(27)-(28)	Вычисление лингвистического коэффициента уверенности и лингвистической векторной оценки категорий	Процесс «Гибридизация»
(29)-(30)	Многозадачное обучение с вычислением адаптивных весов гибридизации	Процесс «Гибридизация»

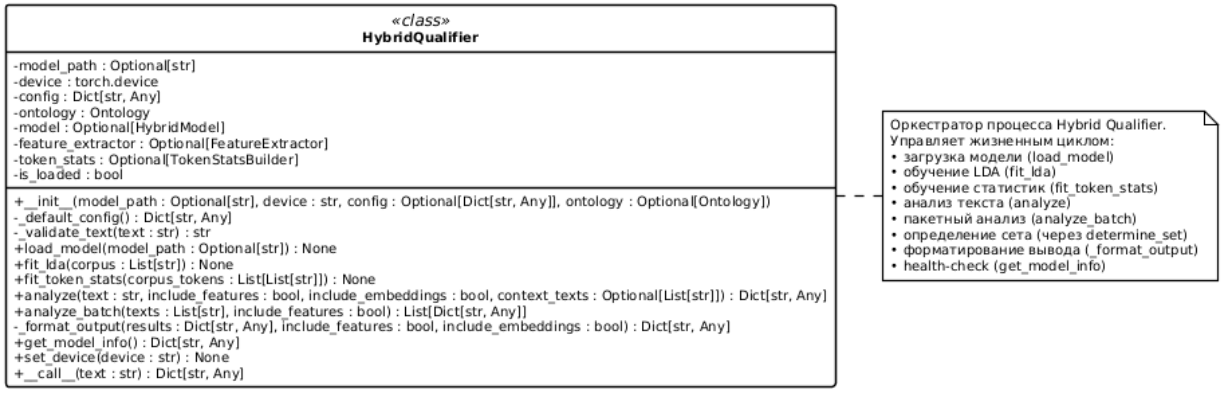
(31)-(32)	Преобразование гибридной оценки в распределение вероятностей и определение итоговой категории	Процесс «Квалификация»
(33)-(34)	Бинарная классификация сарказма с определением итогового флага наличия сарказма	Процесс «Квалификация»
(35)-(36)	Формирование входного вектора и вычисление непрерывной скалярной оценки тяжести	Процесс «Оценка тяжести»
(37)-(39)	Определение обучаемых порогов порядковой регрессии и вычисление вектора квадратичных логитов	Процесс «Оценка тяжести»»
(40)-(41)	Преобразование логитов в распределение вероятностей и определение итогового уровня тяжести	Процесс «Оценка тяжести»

ПРИЛОЖЕНИЕ В
Визуальное представление отдельных классов библиотеки квалификации

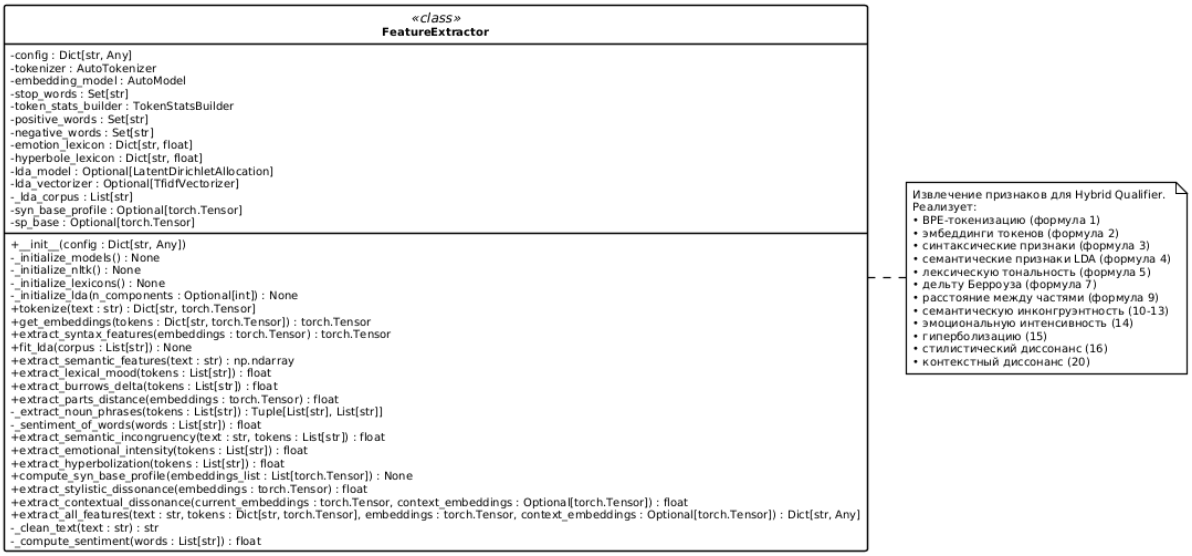
Класс «Hybrid model»



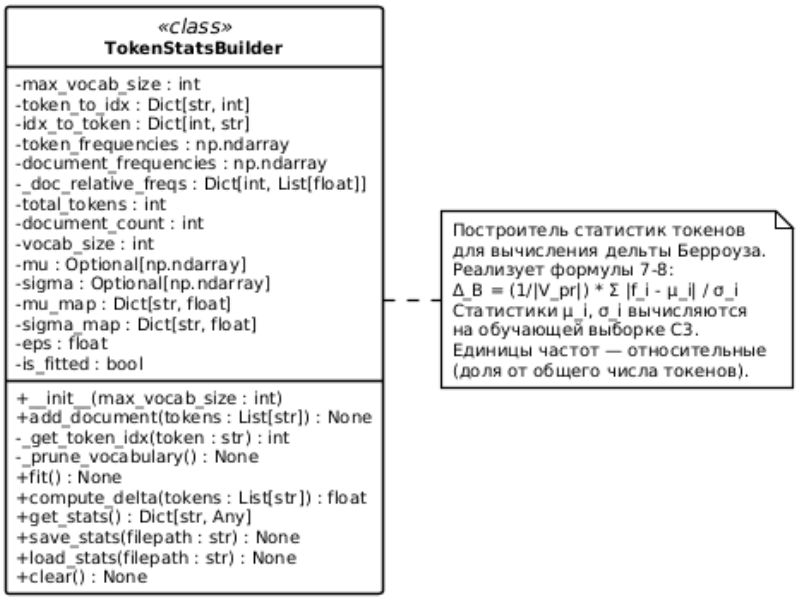
Класс «Hybrid qualifier»



Класс «Feature extractor»



Класс «Token stats builder»



ПРИЛОЖЕНИЕ Г
Программный код подключаемого модуля.

Структура библиотеки

```
hybrid-qualifier/
├── src/
│   └── hybrid_qualifier/
│       ├── __init__.py          # Экспорт публичного API
│       ├── version.py           # Версионирование
│       ├── api/
│       │   ├── __init__.py      # API модули
│       │   ├── client.py        # HTTP клиент
│       │   └── schemas.py       # Схемы данных
│       ├── inference/
│       │   └── base.py          # HybridQualifier (оркестрация формул 1-42)
```

Формула (1) - Токенизация BPE (через FeatureExtractor.tokenize)
Формула (2) - Эмбеддинги токенов (через FeatureExtractor.get_embeddings)
Формулы (3-16) - Извлечение признаков (через FeatureExtractor.extract_all_features)
Формула (20) - Контекстный диссонанс (через context_texts в analyze)
Формулы (21-23) - Семантическое пространство и конвергенция (в HybridModel.forward)
Формулы (24-25) - Интегральный вектор и нормализация (в HybridModel.forward)
Формулы (26-32.1) - Гибридная категоризация (в HybridModel.forward)
Формулы (33-34) - Бинарная классификация сарказма (в HybridModel.forward)
Формулы (35-41) - Порядковая регрессия тяжести (в HybridModel.forward)
Формула (42) - Адаптация коротких сообщений (в HybridModel.adapt_short)
Правило формирования сета - determine_set (вызов в analyze)

```
│       ├── models/
│       │   └── pretrained.py    # Загрузка/сохранение моделей, статистик нормализации
│       ├── core/
│       │   ├── __init__.py      # Core модули
│       │   ├── model.py         # HybridModel (математическая модель)
```

Формула (2) - Эмбеддинги токенов (self.embedding)
Формула (3) - Синтаксические признаки CNN (self.syntax_cnn, self.syntax_fc)
Формула (4) - Семантические признаки LDA (self.lda_transform)
Формула (5) - Лексическая тональность (H_lex)
Формула (6) - Оценка тональности (self.tonality_fc)
Формула (7) - Дельта Берроуза (метод _compute_burrows_delta)
Формула (8) - Нормированная аномальность (delta = sigmoid(...))
Формула (9) - Косинусное расстояние между частями (метод _compute_parts_distance)
Формулы (10-13) - Индекс семантической инконгруэнтности (метод _compute_semantic_incongruency)
Формула (14) - Коэффициент эмоциональной интенсивности (метод _compute_emotional_intensity)
Формула (15) - Индекс гиперболизации (метод _compute_hyperbolization)
Формула (16) - Коэффициент стилистического диссонанса (метод _compute_stylistic_dissonance)
Формула (17) - Веса внимания (Q, K, V, alpha_sarc)
Формула (18) - Вектор контекстной противоречивости (CCV)
Формула (19) - Скаляр саркастичности (rho)
Формула (20) - Коэффициент контекстного диссонанса (метод _compute_contextual_dissonance)
Формула (21) - Семантическое пространство (SP)

Формула (22) - Коэффициент конвергенции (K_conv = cosine_similarity)
Формула (23) - Нормализация семантического пространства (SP_norm)
Формула (24) - Интегральный вектор признаков (priz_sarc)
Формула (25) - Нормализация признаков (priz_norm через mu_priz, sigma_priz)
Формула (26) - Нейросетевая оценка категорий (self.category_fc)
Формула (27) - Лингвистический коэффициент уверенности (kappa)
Формула (28) - Лингвистическая векторная оценка (self.ling_proj)
Формула (29) - Многозадачное обучение (combined_out с self.wp)
Формула (30) - Адаптивные веса (alpha_w = softmax)
Формула (31) - Вероятностное распределение (cat_probs)
Формула (32) - Итоговая категория (category = argmax)
Формула (32.1) - Многометочный режим (сигмоида)
Формулы (33-34) - Бинарная классификация сарказма (self.sarcasm_cls)
Формулы (35-41) - Оценка степени тяжести (self.severity_fc, self.severity_thresholds)
Формула (42) - Адаптация коротких сообщений (метод adapt_short)
Формула (33) - Предикат допустимости сета (функция adm)
Формула (34) - Выбор максимального сета (determine_set)
Формула (34.1) - Выбор по суммарной вероятности (determine_set)
Формула (35) - Уверенность сета (SetResult.confidence)
Формула (36) - Сет «Прочее» (SetResult, determine_set)
│ │ └── **features.py # Извлечение признаков**
Формула (1) - Токенизация BPE (метод tokenize)
Формула (2) - Эмбеддинги токенов (метод get_embeddings)
Формула (3) - Синтаксические признаки (метод extract_syntax_features)
Формула (4) - Семантические признаки LDA (методы fit_lda, extract_semantic_features)
Формула (5) - Лексическая тональность (метод extract_lexical_mood)
Формула (7) - Дельта Берроуза (метод extract_burrows_delta)
Формула (9) - Косинусное расстояние между частями (метод extract_parts_distance)
Формулы (10-13) - Семантическая инконгруэнтность (методы _extract_noun_phrases, _sentiment_of_words, extract_semantic_incongruency)
Формула (14) - Эмоциональная интенсивность (метод extract_emotional_intensity)
Формула (15) - Гиперболизация (метод extract_hyperbolization)
Формула (16) - Стилистический диссонанс (методы compute_syn_base_profile, extract_stylistic_dissonance)
Формула (20) - Контекстный диссонанс (метод extract_contextual_dissonance)
│ └── utils/
│ ├── exceptions.py # Исключения
│ └── **token_stats_builder.py # Построение статистик**
Формула (7) - Дельта Берроуза (метод compute_delta)
Формула (8) - Статистики μ_i, σ_i (метод fit)
Формула (25) - Статистики нормализации признаков (get_stats, save_stats, load_stats)
├── requirements.txt # Зависимости (включая scikit-learn)
└── setup.py # Установка

src/hybrid_qualifier/inference/base.py

```
"""
Основной класс для вывода Hybrid Qualifier.
Оркестрирует формулы 1-42: токенизацию, извлечение признаков,
прямой проход модели, определение сета и форматирование результата.
"""
from typing import Optional, List, Dict, Any
import re
```

```
import time
import torch
from hybrid_qualifier.core.model import (
    HybridModel, SetResult, determine_set, Ontology)
from hybrid_qualifier.core.features import FeatureExtractor
from hybrid_qualifier.models.pretrained import load_pretrained_model
from hybrid_qualifier.utils.token_stats_builder import TokenStatsBuilder
from hybrid_qualifier.utils.exceptions import (
    ModelNotLoadedError,
    InvalidInputError,
    InferenceError)
class HybridQualifier:
    def __init__(
        self,
        model_path: Optional[str] = None,
        device: str = "cpu",
        config: Optional[Dict[str, Any]] = None,
        ontology: Optional[Ontology] = None    ):
        """
        Инициализация HybridQualifier.
        Args:
            model_path: Путь к предобученной модели
            device: Устройство выполнения ('cpu' или 'cuda')
            config: Словарь конфигурации
            ontology: Онтологическая спецификация для определения сета
        """
        self.model_path = model_path
        self.device = torch.device(
            device if torch.cuda.is_available() else "cpu")
        self.config = config or self._default_config()
        self.ontology = ontology or Ontology()
        self.model: Optional[HybridModel] = None
        self.feature_extractor: Optional[FeatureExtractor] = None
        self.token_stats: Optional[TokenStatsBuilder] = None
        self.is_loaded = False
    def _default_config(self) -> Dict[str, Any]:
        """Возвращает конфигурацию по умолчанию."""
        return {
            "embedding_dim": 768,
            "hidden_dim": 256,
            "num_categories": 10,
            "num_severity_levels": 5,
            "num_sentiment_classes": 3,
            "vocab_size": 30522,
            "num_topics": 5,
            "mood_dim": 6,
            "max_seq_length": 512,
            "batch_size": 32,
            "learning_rate": 1e-4,
            "dropout_rate": 0.1,
            "attention_heads": 8,
            "gamma": 1.0,
```

```
        "tau_e": 0.5,
        "context_window": 5,
        "short_text_threshold": 20,
        "short_text_tau": 10,
        "lambda_reg": 0.1,
        "epsilon": 1e-8,
        "max_text_length": 10000        }
    def _validate_text(self, text: str) -> str:
        """
        Валидация и очистка входного текста.
        Args:
            text: Входной текст
        Returns:
            Очищенный текст
        Raises:
            InvalidInputError: Если текст невалидный
        """
        if not text:
            raise InvalidInputError("Текст не может быть пустым")
        if not isinstance(text, str):
            raise InvalidInputError("Текст должен быть строкой")
        max_length = self.config.get('max_text_length', 10000)
        if len(text) > max_length:
            text = text[:max_length]
        text = re.sub(r'\s+', ' ', text)
        text = re.sub(r'[^\w\s.,!?\'"()-]', '', text)
        text = text.strip()
        if not text:
            raise InvalidInputError(
                "Текст содержит только недопустимые символы")
        return text
    def load_model(self, model_path: Optional[str] = None) -> None:
        """
        Загрузка предобученной модели.
        Args:
            model_path: Путь к весам модели (опционально)
        Raises:
            ModelNotLoadedError: Если загрузка не удалась
        """
        try:
            path = model_path or self.model_path
            if path:
                self.model = load_pretrained_model(
                    path, self.device, self.config)
            else:
                self.model = HybridModel(self.config)
                self.model.to(self.device)
            self.feature_extractor = FeatureExtractor(self.config)
            self.token_stats = TokenStatsBuilder()
            self.is_loaded = True
        except Exception as e:
            raise ModelNotLoadedError(f"Не удалось загрузить модель: {e}")
```

```
def fit_lda(self, corpus: List[str]) -> None:
    """Обучение LDA на корпусе (формула 4)."""
    if self.feature_extractor is None:
        raise ModelNotLoadedError("Модель не загружена")
    self.feature_extractor.fit_lda(corpus)
def fit_token_stats(self, corpus_tokens: List[List[str]]) -> None:
    """Обучение статистик токенов для дельты Берроуза (формула 7)."""
    if self.token_stats is None:
        self.token_stats = TokenStatsBuilder()
    for tokens in corpus_tokens:
        self.token_stats.add_document(tokens)
    self.token_stats.fit()
    if self.feature_extractor is not None:
        self.feature_extractor.token_stats_builder = self.token_stats
def analyze(
    self,
    text: str,
    include_features: bool = False,
    include_embeddings: bool = False,
    context_texts: Optional[List[str]] = None
) -> Dict[str, Any]:
    """
    Анализ одного текста.
    Args:
        text: Входной текст для анализа
        include_features: Включать векторы признаков
        include_embeddings: Включать эмбеддинги
        context_texts: Предыдущие сообщения диалога (формула 20)
    Returns:
        Словарь с результатами анализа
    """
    start_time = time.time()
    if not self.is_loaded:
        raise ModelNotLoadedError(
            "Модель не загружена. Вызовите load_model() сначала.")
    text = self._validate_text(text)
    try:
        tokens = self.feature_extractor.tokenize(text)
        embeddings = self.feature_extractor.get_embeddings(tokens)
        context_embeddings = None
        if context_texts:
            ctx_tokens = self.feature_extractor.tokenize(
                " ".join(context_texts))
            context_embeddings = (
                self.feature_extractor.get_embeddings(ctx_tokens))
        features = self.feature_extractor.extract_all_features(
            text, tokens, embeddings, context_embeddings)
        features['context_embeddings'] = context_embeddings
        with torch.no_grad():
            results = self.model.predict(
                features,
                tokens['input_ids'],
```

```
            embeddings,
            include_features=include_features)
        results['processing_time'] = time.time() - start_time
        cat_probs = {
            i: p for i, p in enumerate(results.get('cat_probs', []))}
        set_result = determine_set(
            [results['category']], self.ontology, cat_probs)
        results['set'] = {
            'id': set_result.id,
            'name': set_result.name,
            'confidence': set_result.confidence,
            'requires_expert_review': set_result.requires_expert_review          }
        return self._format_output(
            results, include_features, include_embeddings)
    except Exception as e:
        raise InferenceError(f"Ошибка вывода: {e}")
def analyze_batch(
    self,
    texts: List[str],
    include_features: bool = False
) -> List[Dict[str, Any]]:
    """
    Анализ нескольких текстов в пакете.
    Args:
        texts: Список входных текстов
        include_features: Включать векторы признаков
    Returns:
        Список словарей с результатами анализа
    """
    if not self.is_loaded:
        raise ModelNotLoadedError(
            "Модель не загружена. Вызовите load_model() сначала.")
    if not texts:
        raise InvalidInputError("Список текстов не может быть пустым")
    results = []
    for text in texts:
        try:
            result = self.analyze(text, include_features)
            results.append(result)
        except Exception as e:
            results.append({
                "error": str(e),
                "success": False,
                "text": text[:100] + "..."              })
    return results
def _format_output(
    self,
    results: Dict[str, Any],
    include_features: bool,
    include_embeddings: bool
) -> Dict[str, Any]:
    """Форматирование вывода модели для ответа API."""
```

```
    output = {
        "category": results.get("category", "unknown"),
        "category_prob": results.get("category_prob", 0.0),
        "sarcasm": results.get("sarcasm", False),
        "sarcasm_prob": results.get("sarcasm_prob", 0.0),
        "severity": results.get("severity", 0),
        "severity_probs": results.get("severity_probs", []),
        "sentiment": results.get("sentiment", "neutral"),
        "sentiment_probs": results.get("sentiment_probs", []),
        "token_count": results.get("token_count", 0),
        "processing_time": results.get("processing_time", 0.0),
        "set": results.get("set", {}),
        "success": True        }
    if include_features and "features" in results:
        features = results["features"]
        output["features"] = (
            features.tolist() if hasattr(features, "tolist")
            else features)
    if include_embeddings and "embeddings" in results:
        embeddings = results["embeddings"]
        output["embeddings"] = (
            embeddings.tolist() if hasattr(embeddings, "tolist")
            else embeddings)
    return output
def get_model_info(self) -> Dict[str, Any]:
    """Получение информации о загруженной модели."""
    if not self.is_loaded:
        return {"loaded": False}
    return {
        "loaded": True,
        "device": str(self.device),
        "config": self.config,
        "categories": (
            self.model.category_names
            if hasattr(self.model, 'category_names') else []),
        "severity_levels": (
            self.model.severity_names
            if hasattr(self.model, 'severity_names') else []),
        "sentiment_classes": ["negative", "neutral", "positive"]        }
def set_device(self, device: str) -> None:
    """Установка устройства для вывода."""
    self.device = torch.device(
        device if torch.cuda.is_available() else "cpu")
    if self.model:
        self.model.to(self.device)
def __call__(self, text: str) -> Dict[str, Any]:
    """Разрешает вызов экземпляра как функции."""
    return self.analyze(text)
```

src/hybrid_qualifier/core/model.py

```
"""
Математическая модель Hybrid Qualifier.
Реализует фреймворк, включающий:
- Эмбеддинги токенов (формула 2)
- Синтаксические признаки (формула 3)
- Семантические признаки (формула 4)
- Лексическую и общую тональность (формулы 5-6)
- Дельту Берроуза и её нормировку (формулы 7-8)
- Признаки сарказма (формулы 9-16)
- Механизм внимания (формулы 17-19)
- Контекстный диссонанс (формула 20)
- Семантическое пространство и конвергенцию (формулы 21-23)
- Интегральный вектор признаков и его нормализацию (формулы 24-25)
- Гибридную категоризацию (формулы 26-32.1)
- Правило формирования сета (формулы 33-36)
- Бинарную классификацию сарказма (формулы 33-34)
- Порядковую регрессию тяжести (формулы 35-41)
- Адаптацию коротких сообщений (формула 42)
"""
import math
from dataclasses import dataclass, field
from typing import Optional, Dict, Any, Tuple, List, Callable
import torch
import torch.nn as nn
import torch.nn.functional as F
import numpy as np
@dataclass
class SetResult:
    """Результат определения сета."""
    id: int
    name: str
    confidence: float
    categories: List[int]
    requires_expert_review: bool = False
@dataclass
class OntologySet:
    """Описание сета онтологии."""
    id: int
    name: str
    categories: List[int]
@dataclass
class Ontology:
    """
    Онтологическая спецификация Ω = (A, Θ).
    A - множество аксиом: α(S, Cat) -> bool.
    Θ - множество правил: θ(S, Cat) -> bool.
    sets - множество рассматриваемых сетов.
    """
    sets: List[OntologySet] = field(default_factory=list)
    axioms: List[Callable[[OntologySet, List[int]], bool]] = field(
        default_factory=list)
```

```
    rules: List[Callable[[OntologySet, List[int]], bool]] = field(
        default_factory=list)
def adm(
    s: OntologySet,
    cat: List[int],
    omega: Ontology
) -> int:
    """
    Формула 33: предикат допустимости сета.
    Adm(S, Cat, Ω) = 1[Cat ⊆ S.cat]
                   * 1[∀α ∈ A: α(S, Cat) = true]
                   * 1[∀θ ∈ Θ: θ(S, Cat) = true]
    """
    cat_set = set(cat)
    if not cat_set <= set(s.categories):
        return 0
    for axiom in omega.axioms:
        if not axiom(s, cat):
            return 0
    for rule in omega.rules:
        if not rule(s, cat):
            return 0
    return 1
def determine_set(
    categories: List[int],
    ontology: Ontology,
    cat_probs: Optional[Dict[int, float]] = None
) -> SetResult:
    """
    Правило формирования сета: Set = f(Cat, Ω).
    Формула 34: Set = argmax_{S ∈ Sets} Σ_{c ∈ Cat} 1[c ∈ S.cat] * Adm(S, Cat, Ω)
    Формула 34.1: при равной мощности - argmax Σ_{c ∈ Cat ∩ S.cat} cat_prob[c] * Adm
    Формула 35: conf(Set) = min_{c ∈ Cat ∩ S.cat} cat_prob[c]
    Формула 36: если ∀S ∈ Sets: Adm(S, Cat, Ω) = 0 → сет «Прочее»
    """
    cat_probs = cat_probs or {}
    cat_set = set(categories)
    best: Optional[OntologySet] = None
    best_coverage = -1
    best_prob_sum = -1.0
    any_admissible = False
    for s in ontology.sets:
        if adm(s, categories, ontology) == 0:
            continue
        any_admissible = True
        coverage = sum(1 for c in cat_set if c in s.categories)
        prob_sum = sum(
            cat_probs.get(c, 0.0) for c in cat_set if c in s.categories)
        if coverage > best_coverage or (
            coverage == best_coverage and prob_sum > best_prob_sum
        ):
            best = s
```

```
            best_coverage = coverage
            best_prob_sum = prob_sum
    if not any_admissible or best is None:
        return SetResult(
            id=0,
            name="Прочее",
            confidence=0.0,
            categories=list(categories),
            requires_expert_review=True)
    covered = [c for c in cat_set if c in best.categories]
    confidence = min(cat_probs.get(c, 1.0) for c in covered)
    return SetResult(
        id=best.id,
        name=best.name,
        confidence=float(confidence),
        categories=list(categories),
        requires_expert_review=False)
class HybridModel(nn.Module):
    def __init__(self, config: Dict[str, Any]):
        super().__init__()
        self.config = config
        self._init_params()
        self._build_layers()
        self._init_weights()
        self._init_auxiliary_weights()
        # Статистики обучающей выборки (формулы 23, 25)
        self.register_buffer('mu_priz', torch.zeros(self.D_sarc))
        self.register_buffer('sigma_priz', torch.ones(self.D_sarc))
        self.register_buffer('mu_sp', torch.zeros(self.embedding_dim))
        self.register_buffer('sigma_sp', torch.ones(self.embedding_dim))
        # Формула 22: базовое семантическое пространство SP_base
        self.register_buffer('sp_base', torch.zeros(self.embedding_dim))
        # Формула 16: эталонный синтаксический профиль H_syn^base(k)
        self.register_buffer('syn_base_profile', torch.zeros(self.h_syn))
        # Формула 38: граничные пороги θ_0 и θ_4
        self.register_buffer('theta_neg_inf', torch.tensor(-1000.0))
        self.register_buffer('theta_pos_inf', torch.tensor(1000.0))
        self._init_feature_groups()
    def _init_params(self) -> None:
        """Инициализация параметров модели из конфигурации."""
        self.embedding_dim = self.config.get('embedding_dim', 768)
        self.hidden_dim = self.config.get('hidden_dim', 256)
        self.num_categories = self.config.get('num_categories', 10)
        self.num_severity_levels = self.config.get('num_severity_levels', 5)
        self.num_sentiment_classes = self.config.get(
            'num_sentiment_classes', 3)
        self.vocab_size = self.config.get('vocab_size', 30522)
        self.num_topics = self.config.get('num_topics', 5)
        self.mood_dim = self.config.get('mood_dim', 6)
        self.max_seq_length = self.config.get('max_seq_length', 512)
        self.attention_heads = self.config.get('attention_heads', 8)
        self.gamma = self.config.get('gamma', 1.0)
```

```
        self.tau_e  = self.config.get('tau_e',  0.5)
        self.lambda_reg  = self.config.get('lambda_reg',  0.1)
        self.epsilon  = self.config.get('epsilon',  1e-8)
        self.short_text_threshold  = self.config.get(
            'short_text_threshold',  20)
        self.short_text_tau  = self.config.get('short_text_tau',  10)
        self.d_k  = self.embedding_dim  // self.attention_heads
        self.d_v  = self.embedding_dim
        self.h_syn  = self.hidden_dim
        self.h_sem  = self.num_topics
        self.d_mood  = self.mood_dim
        # Формула 24: D_sarc = h_syn + h_sem + d_
        self.D_sarc  = (
            self.h_syn  + self.h_sem  + self.d_
        # Формула 35: D_sev = (h_syn  + h_sem)
        self.D_sev  = (
            (self.h_syn  + self.h_sem)  + self.embedding_dim)
    def _build_layers(self)  -> None:
        """Построение  слоёв нейронной сети."""
        # Формула 2: слой эмбеддингов
        self.embedding  = nn.Embedding(
            self.vocab_size,  self.embedding_dim,  padding_idx=0)
        # Формула 3: синтаксические CNN-признаки
        self.syntax_cnn  = nn.ModuleList([
            nn.Conv1d(self.embedding_dim,  self.hidden_dim,  kernel_size=k)
            for k in [2, 3, 4, 5]])
        self.syntax_fc  = nn.Linear(
            self.hidden_dim  * len([2,  3, 4, 5]), self.h_syn)
        # Формула 4: проекция семантических признаков
        self.lda_transform  = nn.Linear(self.h_sem,  self.h_sem)
        # Формула 6: сеть тональности
        self.tonality_fc  = nn.Linear(self.h_sem  + self.d_mood,  3)
        # Формула 17: механизм внимания
        self.W_q  = nn.Linear(5,  self.d_k)
        self.W_k  = nn.Linear(self.embedding_dim,  self.d_k)
        self.W_v  = nn.Linear(self.embedding_dim,  self.d_v)
        # Формула 19: скаляр саркастичности
        self.sarcasm_fc  = nn.Linear(self.d_v,  1)
        # Формула 26: классификация категорий
        self.category_fc  = nn.Linear(self.D_sarc,  self.num_categories)
        # Формула 28: лингвистическая проекция
        self.ling_proj  = nn.Linear(self.embedding_dim,  self.num_categories)
        # Формула 29: многозадачное обучение
        self.wp  = nn.Parameter(torch.ones(self.num_categories))
        # Бинарная классификация сарказма
        self.sarcasm_cls  = nn.Linear(self.D_sarc,  2)
        # Формула 36: оценка тяжести
        self.severity_fc  = nn.Linear(self.D_sev,  1)
        # Формула 37: обучаемые пороги порядковой регрессии
        self.severity_thresholds  = nn.Parameter(
            torch.tensor([-2.0,  -1.0, 1.0, 2.0], dtype=torch.float32))
        # Формула 30: адаптивные веса гибридизации
```

```
    self.alpha  = nn.Parameter(torch.tensor(1.0))
    self.beta  = nn.Parameter(torch.tensor(1.0))
    self.gamma_w  = nn.Parameter(torch.tensor(1.0))
def _init_weights(self)  -> None:
    """Инициализация весов Xavier."""
    for module  in self.modules():
        if isinstance(module,  nn.Linear):
            nn.init.xavier_uniform_(module.weight)
            if module.bias  is not None:
                nn.init.zeros_(module.bias)
        elif isinstance(module,  nn.Conv1d):
            nn.init.xavier_uniform_(module.weight)
            if module.bias  is not None:
                nn.init.zeros_(module.bias)
        elif isinstance(module,  nn.Embedding):
            nn.init.normal_(module.weight,  mean=0, std=0.02)
def _init_auxiliary_weights(self)  -> None:
    """Инициализация вспомогательных весов."""
    # Формулы 10-13: проекция тональности
    self.sentiment_proj  = nn.Linear(self.embedding_dim,  1)
def _init_feature_groups(self)  -> None:
    """
    Разделение интегрального вектора на локальные (L) и глобальные (G)
    группы признаков для формулы  42.
    Порядок в priz_sarc:  [H_syn,  H_sem,  CCV, rho, J_sem,  E_int,
    D_parts,  H_hyp,  S_dis,  C_dis].
    L = [H_syn,  CCV, rho, J_sem,  E_int,  D_parts,  H_hyp,  S_dis,  C_dis]
    G = [H_sem]
    """
    self._local_dims  = (
        self.h_syn  + self.d_v)
    self._global_dims  = self.h_sem
def _split_features(
    self,  priz:  torch.Tensor
) -> Tuple[torch.Tensor,  torch.Tensor]:
    """Разделение  priz_sarc  на локальные и глобальные признаки."""
    local  = torch.cat([
        priz[:,  :self.h_syn],
        priz[:,  self.h_syn  + self.h_sem:
                 self.h_syn  + self.h_sem  + self.d_v],
        priz[:,  self.h_syn  + self.h_sem  + self.d_v:]
    ], dim=1)
    global_  = priz[:,  self.h_syn:self.h_syn  + self.h_sem]
    return local,  global_
def adapt_short(
    self,
    text_len:  int,
    priz_local:  torch.Tensor,
    priz_global:  torch.Tensor
) -> torch.Tensor:
    """
    Формула 42: адаптация коротких сообщений.
```

```
    Priz_adapted = [α(n)·Priz_L ; β(n)·Priz_G],
    α(n)=σ((n-n_0)/τ), β(n)=1-α(n), n_0=20, τ=10
    """
    n0 = self.short_text_threshold
    tau = self.short_text_tau
    alpha = torch.sigmoid(
        torch.tensor((n0 - text_len) / tau, dtype=torch.float32,
                     device=priz_local.device))
    beta = 1.0 - alpha
    return torch.cat([alpha * priz_local, beta * priz_global], dim=-1)
def set_calibration(
    self,
    mu_priz: torch.Tensor,
    sigma_priz: torch.Tensor,
    mu_sp: torch.Tensor,
    sigma_sp: torch.Tensor,
    sp_base: torch.Tensor,
    syn_base_profile: torch.Tensor
) -> None:
    """
    Установка статистик обучающей выборки и эталонных профилей.
    Значения должны быть вычислены на обучающей выборке и сохранены
    в артефактах модели.
    """
    self.mu_priz.copy_(mu_priz)
    self.sigma_priz.copy_(sigma_priz)
    self.mu_sp.copy_(mu_sp)
    self.sigma_sp.copy_(sigma_sp)
    self.sp_base.copy_(sp_base)
    self.syn_base_profile.copy_(syn_base_profile)
def forward(
    self,
    input_ids: torch.Tensor,
    attention_mask: torch.Tensor,
    token_stats: Dict,
    context_embeddings: Optional[torch.Tensor] = None,
    mood_embeddings: Optional[torch.Tensor] = None,
    hyper_weights: Optional[torch.Tensor] = None,
    semantic_features: Optional[torch.Tensor] = None
) -> Dict[str, torch.Tensor]:
    """        Прямой проход через гибридную модель (формулы 2-42).        """
    batch_size, seq_len = input_ids.shape
    device = input_ids.device
    # Формула 2: X = E[T]
    X = self.embedding(input_ids)
    # Формула 3: H_syn = CNN_syn(X)
    X_permuted = X.permute(0, 2, 1).contiguous()
    conv_outputs = []
    for conv in self.syntax_cnn:
        conv_out = conv(X_permuted)
        pooled = torch.max(conv_out, dim=-1)[0]
        conv_outputs.append(pooled)
```

```
H_syn = torch.cat(conv_outputs, dim=1)
H_syn = torch.tanh(self.syntax_fc(H_syn))
# Формула 4: H_sem = LDA(S)
if not isinstance(semantic_features, torch.Tensor):
    semantic_features = torch.as_tensor(
        semantic_features, dtype=torch.float32, device=device)
H_sem = semantic_features.to(device)
if H_sem.dim() == 1:
    H_sem = H_sem.unsqueeze(0).expand(batch_size, -1)
H_sem = torch.softmax(self.lda_transform(H_sem), dim=1)
# Формула 5: b_ton - средняя лексическая тональность
if mood_embeddings is None:
    raise ValueError(
        "mood_embeddings обязательны для формулы 5")
H_lex = mood_embeddings.mean(dim=1)
# Формула 6: H_ton = softmax(W_ton · [H_sem ⊕ H_lex] + b_ton)
tonality_input = torch.cat([H_sem, H_lex], dim=1)
H_ton = torch.softmax(self.tonality_fc(tonality_input), dim=1)
# Формула 7: Δ_B
delta_B = self._compute_burrows_delta(input_ids, token_stats)
# Формула 8: δ = σ(Δ_B / γ)
delta = torch.sigmoid(delta_B / (self.gamma + self.epsilon))
# Формула 9: D_parts = 1 - cos(Emb_first, Emb_last)
D_parts = self._compute_parts_distance(X)
# Формулы 10-13: J_sem = |sent_topic - sent_context|
J_sem = self._compute_semantic_incongruency(X)
# Формула 14: E_int
E_int = self._compute_emotional_intensity(X)
# Формула 15: H_hyp
if hyper_weights is None:
    raise ValueError(
        "hyper_weights обязательны для формулы 15")
H_hyp = self._compute_hyperbolization(hyper_weights)
# Формула 16: S_dis
S_dis = self._compute_stylistic_dissonance(H_syn)
# Формула 17: α_i^sarc = softmax((Q · K_i^T)/√d_k)
features_stack = torch.stack(
    [J_sem, E_int, D_parts, H_hyp, S_dis], dim=1)
Q = self.W_q(features_stack)
K = self.W_k(X)
V = self.W_v(X)
attention_scores = torch.matmul(
    Q.unsqueeze(1), K.transpose(1, 2)) / math.sqrt(self.d_k)
mask = attention_mask.unsqueeze(1) == 0
attention_scores = attention_scores.masked_fill(mask, float('-inf'))
alpha_sarc = torch.softmax(attention_scores, dim=-1)
# Формула 18: CCV = Σ α_i^sarc · V_i
CCV = torch.matmul(alpha_sarc, V).squeeze(1)
# Формула 19: ρ = σ(W_ρ · CCV + b_ρ)
rho = torch.sigmoid(self.sarcasm_fc(CCV)).squeeze(-1)
# Формула 20: C_dis = 1 - cos(SP_current, SP_context)
if context_embeddings is None:
```

```
        raise ValueError(
            "context_embeddings обязательны для формулы 20")
    C_dis = self._compute_contextual_dissonance(X, context_embeddings)
    # Формула 21: SP = Σ a_i · X_i
    a_i = torch.softmax(X.mean(dim=-1), dim=-1)
    SP = torch.matmul(a_i.unsqueeze(1), X).squeeze(1)
    # Формула 22: K = cos(SP_new, SP_base)
    K_conv = F.cosine_similarity(
        SP, self.sp_base.to(device).unsqueeze(0), dim=1)
    # Формула 23: SP_norm = (SP - μ_SP) / (σ_SP + ε)
    sigma_sp_safe = torch.where(
        self.sigma_sp == 0,
        torch.ones_like(self.sigma_sp),
        self.sigma_sp)
    SP_norm = (SP - self.mu_sp.to(device)) / (
        sigma_sp_safe.to(device) + self.epsilon)
    # Формула 24: Priz_sarc_new
    priz_sarc = torch.cat([
        H_syn, H_sem, CCV, rho.unsqueeze(1),
        J_sem.unsqueeze(1), E_int.unsqueeze(1),
        D_parts.unsqueeze(1), H_hyp.unsqueeze(1),
        S_dis.unsqueeze(1), C_dis.unsqueeze(1)
    ], dim=1)
    priz_sarc = priz_sarc * (1 + self.lambda_reg * delta.unsqueeze(1))
    # Формула 42: адаптация коротких сообщений
    priz_local, priz_global = self._split_features(priz_sarc)
    priz_sarc = self.adapt_short(seq_len, priz_local, priz_global)
    # Формула 25: Priz_norm = (Priz_sarc_new - μ_Priz) / (σ_Priz + ε)
    sigma_priz_safe = torch.where(
        self.sigma_priz == 0,
        torch.ones_like(self.sigma_priz),
        self.sigma_priz)
    priz_norm = (priz_sarc - self.mu_priz.to(device)) / (
        sigma_priz_safe.to(device) + self.epsilon)
    # Формула 26: neural_out = softmax(W_cat · Priz_norm + b_cat)
    neural_out = torch.softmax(self.category_fc(priz_norm), dim=1)
    # Формула 27: κ = σ(K)
    kappa = torch.sigmoid(K_conv)
    # Формула 28: ling_vect = W_proj · (SP_norm ⊙ κ)
    ling_vect = self.ling_proj(SP_norm) * kappa.unsqueeze(1)
    # Формула 30: α', β', γ' = softmax([α, β, γ])
    alpha_w = torch.softmax(
        torch.stack([self.alpha, self.beta, self.gamma_w]), dim=0)
    alpha_prime, beta_prime, gamma_prime = (
        alpha_w[0], alpha_w[1], alpha_w[2])
    # Формула 29: combined_out
    combined_out = (
        alpha_prime * neural_out
        + beta_prime * ling_vect
        + gamma_prime * (rho.unsqueeze(1) * self.wp.unsqueeze(0)))
    # Формула 31: cat_probs = softmax(combined_out)
    cat_probs = torch.softmax(combined_out, dim=1)
```

```
    # Формула 32: category = argmax(cat_probs)
    category = torch.argmax(cat_probs, dim=1)
    # Формулы 33-34: sarc_probs = softmax(W_sarc · Priz_norm + b_sarc)
    sarcasm_logits = self.sarcasm_cls(priz_norm)
    sarc_probs = torch.softmax(sarcasm_logits, dim=1)
    is_sarcasm = torch.argmax(sarc_probs, dim=1)
    # Формулы 35-41: оценка степени тяжести
    sev_input = torch.cat([
        priz_norm[:, :self.h_syn + self.h_sem],
        H_ton,
        delta.unsqueeze(1),
        kappa.unsqueeze(1),
        SP_norm,
        torch.log(1 + delta_B).unsqueeze(1),
        J_sem.unsqueeze(1),
        E_int.unsqueeze(1),
        rho.unsqueeze(1)
    ], dim=1)
    s = self.severity_fc(sev_input).squeeze(-1)
    logits = []
    for k in range(self.num_severity_levels):
        if k == 0:
            theta = self.theta_neg_inf
        elif k == self.num_severity_levels - 1:
            theta = self.theta_pos_inf
        else:
            theta = self.severity_thresholds[k - 1]
        if isinstance(theta, torch.Tensor):
            theta = theta.to(s.device)
        logit = -(s - theta) ** 2
        logits.append(logit)
    L = torch.stack(logits, dim=1)
    sev_probs = torch.softmax(L, dim=1)
    severity = torch.argmax(sev_probs, dim=1)
    return {
        'X': X, 'H_syn': H_syn, 'H_sem': H_sem, 'H_lex': H_lex,
        'H_ton': H_ton, 'delta_B': delta_B, 'delta': delta,
        'D_parts': D_parts, 'J_sem': J_sem, 'E_int': E_int,
        'H_hyp': H_hyp, 'S_dis': S_dis, 'CCV': CCV, 'rho': rho,
        'C_dis': C_dis, 'SP': SP, 'K': K_conv, 'SP_norm': SP_norm,
        'priz_sarc': priz_sarc, 'priz_norm': priz_norm,
        'neural_out': neural_out, 'kappa': kappa,
        'ling_vect': ling_vect, 'combined_out': combined_out,
        'cat_probs': cat_probs, 'category': category,
        'sarc_probs': sarc_probs, 'is_sarcasm': is_sarcasm,
        's': s, 'sev_probs': sev_probs, 'severity': severity        }
def _compute_burrows_delta(
    self,
    input_ids: torch.Tensor,
    token_stats: Dict
) -> torch.Tensor:
    """
```

```
    Формула 7: Δ_B = (1/|V_pr|) * Σ_{i∈V_pr} |f_i - μ_i| / σ_i
    """
    batch_size, seq_len = input_ids.shape
    device = input_ids.device
    mu = token_stats.get('mu')
    sigma = token_stats.get('sigma')
    if mu is None or sigma is None:
        raise ValueError(
            "token_stats должен содержать 'mu' и 'sigma'")
    mu_t = torch.as_tensor(mu, dtype=torch.float32, device=device)
    sigma_t = torch.as_tensor(sigma, dtype=torch.float32, device=device)
    vocab = mu_t.shape[0]
    token_counts = torch.zeros(batch_size, vocab, device=device)
    for b in range(batch_size):
        ids = input_ids[b]
        for i in range(seq_len):
            tid = ids[i].item()
            if 0 <= tid < vocab:
                token_counts[b, tid] += 1
    delta_B = torch.zeros(batch_size, device=device)
    for b in range(batch_size):
        freqs = token_counts[b] / max(seq_len, 1)
        present = (token_counts[b] > 0) & (sigma_t > self.epsilon)
        if present.any():
            diff = torch.abs(freqs[present] - mu_t[present]) / (
                sigma_t[present] + self.epsilon)
            delta_B[b] = diff.mean()
    return delta_B
def _compute_parts_distance(self, X: torch.Tensor) -> torch.Tensor:
    """Формула 9: D_parts = 1 - cos(Emb_first, Emb_last)."""
    batch_size, seq_len, dim = X.shape
    k = max(1, seq_len // 3)
    emb_first = X[:, :k, :].mean(dim=1)
    emb_last = X[:, -k:, :].mean(dim=1)
    cos_sim = F.cosine_similarity(emb_first, emb_last, dim=1)
    return torch.clamp(1.0 - cos_sim, 0.0, 2.0)
def _compute_semantic_incongruency(
    self,
    X: torch.Tensor
) -> torch.Tensor:
    """Формулы 10-13: J_sem = |sent_topic - sent_context|."""
    batch_size, seq_len, dim = X.shape
    device = X.device
    topic_len = max(1, seq_len // 3)
    topic_mask = torch.zeros(batch_size, seq_len, device=device)
    topic_mask[:, :topic_len] = 1
    context_mask = 1 - topic_mask
    topic_emb = (X * topic_mask.unsqueeze(-1)).sum(dim=1) / (
        topic_mask.sum(dim=1, keepdim=True) + self.epsilon)
    context_emb = (X * context_mask.unsqueeze(-1)).sum(dim=1) / (
        context_mask.sum(dim=1, keepdim=True) + self.epsilon)
    sent_topic = torch.tanh(self.sentiment_proj(topic_emb)).squeeze(-1)
```

```
        sent_context = torch.tanh(
            self.sentiment_proj(context_emb)).squeeze(-1)
        return torch.clamp(
            torch.abs(sent_topic - sent_context), 0.0, 2.0)
    def _compute_emotional_intensity(
        self,
        X: torch.Tensor
    ) -> torch.Tensor:
        """Формула 14: E_int."""
        batch_size, seq_len, dim = X.shape
        device = X.device
        if seq_len < 2:
            return torch.zeros(batch_size, device=device)
        std = X.std(dim=-1)
        std_safe = torch.where(std == 0, torch.ones_like(std), std)
        emo_scores = torch.sigmoid(
            X.mean(dim=-1) / (std_safe + self.epsilon))
        diffs = torch.abs(emo_scores[:, 1:] - emo_scores[:, :-1])
        mask = diffs > self.tau_e
        E_int = torch.zeros(batch_size, device=device)
        for b in range(batch_size):
            valid = diffs[b][mask[b]]
            if valid.numel() > 0:
                E_int[b] = valid.mean()
        return E_int
    def _compute_hyperbolization(
        self,
        hyper_weights: torch.Tensor
    ) -> torch.Tensor:
        """Формула 15: H_hyp."""
        H_hyp = hyper_weights.mean(dim=1)
        return torch.clamp(H_hyp, 0.0, 1.0)
    def _compute_stylistic_dissonance(
        self,
        H_syn: torch.Tensor
    ) -> torch.Tensor:
        """Формула 16: S_dis."""
        device = H_syn.device
        base = self.syn_base_profile.to(device)
        diff = torch.norm(H_syn - base.unsqueeze(0), dim=1)
        norm_base = torch.norm(base)
        S_dis = diff / (norm_base + self.epsilon)
        return torch.clamp(S_dis, 0.0, 10.0)
    def _compute_contextual_dissonance(
        self,
        X_current: torch.Tensor,
        context_embeddings: torch.Tensor
    ) -> torch.Tensor:
        """Формула 20: C_dis."""
        SP_current = X_current.mean(dim=1)
        SP_context = context_embeddings.mean(dim=1)
        cos_sim = F.cosine_similarity(SP_current, SP_context, dim=1)
```

```
        return torch.clamp(1.0 - cos_sim, 0.0, 2.0)
    def predict(
        self,
        features: Dict[str, torch.Tensor],
        tokens: torch.Tensor,
        embeddings: torch.Tensor,
        include_features: bool = False
    ) -> Dict[str, Any]:
        """Выполнение предсказания на извлечённых признаках."""
        outputs = self.forward(
            tokens,
            attention_mask=features['attention_mask'],
            token_stats=features['token_stats'],
            context_embeddings=features.get('context_embeddings'),
            mood_embeddings=features.get('mood_embeddings'),
            hyper_weights=features.get('hyper_weights'),
            semantic_features=features['semantic_features'])
        cat_probs_full = outputs['cat_probs'][0].detach().cpu().tolist()
        results = {
            'category': outputs['category'].item(),
            'category_prob': outputs['cat_probs'][0].max().item(),
            'sarcasm': outputs['is_sarcasm'].item() == 1,
            'sarcasm_prob': outputs['sarc_probs'][0, 1].item(),
            'severity': outputs['severity'].item(),
            'severity_probs': outputs['sev_probs'][0].cpu().tolist(),
            'sentiment': self._get_sentiment_label(outputs['H_ton'][0]),
            'sentiment_probs': outputs['H_ton'][0].cpu().tolist(),
            'token_count': tokens.shape[1],
            'cat_probs': cat_probs_full        }
        if include_features:
            results['features'] = (
                outputs['priz_sarc'][0].detach().cpu().numpy())
        return results
    def _get_sentiment_label(self, tonality_probs: torch.Tensor) -> str:
        """Получение метки тональности из вероятностей."""
        idx = torch.argmax(tonality_probs).item()
        labels = ['negative', 'neutral', 'positive']
        return labels[idx] if idx < len(labels) else 'neutral'
```

src/hybrid_qualifier/core/features.py

```
"""
Модуль извлечения признаков для Hybrid Qualifier.
Реализует извлечение признаков, необходимых для модели:
- BPE-токенизация (формула 1)
- Эмбеддинги токенов (формула 2)
- Синтаксические признаки (формула 3)
- Семантические признаки через LDA (формула 4)
- Лексическая тональность (формула 5)
- Дельта Берроуза (формула 7)
- Косинусное расстояние между частями текста (формула 9)
- Семантическая инконгруэнтность (формулы 10-13)
- Эмоциональная интенсивность (формула 14)
```

```
- Гиперболизация (формула 15)
- Стилистический диссонанс (формула 16)
- Контекстный диссонанс (формула 20)
"""
from typing import Dict, Any, Optional, List, Tuple
from collections import Counter
import re
import math
import numpy as np
import torch
import torch.nn.functional as F
import nltk
from nltk.corpus import stopwords
from nltk.corpus import opinion_lexicon
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from transformers import AutoTokenizer, AutoModel
from hybrid_qualifier.utils.token_stats_builder import TokenStatsBuilder
from hybrid_qualifier.utils.exceptions import FeatureExtractionError
class FeatureExtractor:
    """ Извлекает признаки, необходимые для модели Hybrid Qualifier. """
    def __init__(self, config: Dict[str, Any]):
        self.config = config
        self.tokenizer = None
        self.embedding_model = None
        self.stop_words = set()
        self.token_stats_builder = TokenStatsBuilder()
        # Лексиконы для тональности и эмоций
        self.positive_words: set = set()
        self.negative_words: set = set()
        self.emotion_lexicon: Dict[str, float] = {}
        self.hyperbole_lexicon: Dict[str, float] = {}
        # LDA-модель и векторное представление корпуса
        self.lda_model: Optional[LatentDirichletAllocation] = None
        self.lda_vectorizer: Optional[TfidfVectorizer] = None
        self._lda_corpus: List[str] = []
        # Эталонный синтаксический профиль, вычисляемый на обучающей выборке
        self.syn_base_profile: Optional[torch.Tensor] = None
        # Эталонное семантическое пространство (формула 22)
        self.sp_base: Optional[torch.Tensor] = None
        self._initialize_models()
        self._initialize_nltk()
        self._initialize_lexicons()
        self._initialize_lda()
    def _initialize_models(self) -> None:
        """ Инициализация токенизатора и модели эмбеддингов (формулы 1-2). """
        try:
            model_name = self.config.get(
                'embedding_model', 'bert-base-uncased')
            self.tokenizer = AutoTokenizer.from_pretrained(model_name)
            self.embedding_model = AutoModel.from_pretrained(model_name)
            self.embedding_model.eval()
```

```
        except Exception as e:
            raise FeatureExtractionError(
                f"Не удалось инициализировать модели: {e}")
    def _initialize_nltk(self) -> None:
        for resource, path in [
            ('punkt', 'tokenizers/punkt'),
            ('averaged_perceptron_tagger',
             'taggers/averaged_perceptron_tagger'),
            ('stopwords', 'corpora/stopwords'),
            ('opinion_lexicon', 'corpora/opinion_lexicon'),        ]:
            try:
                nltk.data.find(path)
            except LookupError:
                nltk.download(resource, quiet=True)
        self.stop_words = set(stopwords.words('english'))
    def _initialize_lexicons(self) -> None:
        self.positive_words = set(opinion_lexicon.positive())
        self.negative_words = set(opinion_lexicon.negative())
        # NRC-подобный эмоциональный лексикон
        self.emotion_lexicon = {
            'anger': 0.9, 'angry': 0.9, 'furious': 1.0, 'rage': 1.0,
            'joy': 0.9, 'happy': 0.9, 'delight': 0.8, 'love': 0.9,
            'sadness': 0.8, 'sad': 0.8, 'sorrow': 0.9, 'cry': 0.7,
            'fear': 0.8, 'afraid': 0.8, 'scared': 0.8, 'terror': 1.0,
            'surprise': 0.7, 'wow': 0.7, 'shock': 0.8, 'amazed': 0.7,
            'disgust': 0.9, 'disgusting': 0.9, 'revolting': 1.0,
            'trust': 0.6, 'calm': 0.4, 'neutral': 0.0}
        # Лексикон гипербол
        self.hyperbole_lexicon = {
            'always': 0.6, 'never': 0.7, 'everyone': 0.6, 'nobody': 0.7,
            'everything': 0.7, 'nothing': 0.7, 'absolutely': 0.9,
            'totally': 0.8, 'completely': 0.9, 'extremely': 0.9,
            'incredibly': 0.9, 'insanely': 1.0, 'literally': 0.7,
            'forever': 0.7, 'infinity': 0.9, 'million': 0.6,
            'billion': 0.8, 'perfect': 0.8, 'worst': 0.8, 'best': 0.7,
            'huge': 0.6, 'tiny': 0.5, 'giant': 0.6, 'super': 0.7,
            'ultra': 0.8}
    def _initialize_lda(self, n_components: Optional[int] = None) -> None:
        """        Инициализация LDA-модели (формула 4).        """
        n_topics = n_components or self.config.get('num_topics', 5)
        self.lda_model = LatentDirichletAllocation(
            n_components=n_topics,
            random_state=42,
            learning_method='batch',
            max_iter=50)
        self.lda_vectorizer = TfidfVectorizer(
            max_features=self.config.get('lda_max_features', 5000),
            stop_words='english',
            lowercase=True)
    def tokenize(self, text: str) -> Dict[str, torch.Tensor]:
        """        Формула 1: Токенизация текста с использованием BPE.        """
        if not text or not text.strip():
```

```
            raise ValueError("Текст не может быть пустым")
        text = self._clean_text(text)
        tokens = self.tokenizer(
            text,
            truncation=True,
            padding='max_length',
            max_length=self.config.get('max_seq_length', 512),
            return_tensors='pt')
        return {
            'input_ids': tokens['input_ids'],
            'attention_mask': tokens['attention_mask'],
            'token_type_ids': tokens.get('token_type_ids'),
            'tokens': self.tokenizer.convert_ids_to_tokens(
                tokens['input_ids'][0])}
    def get_embeddings(
        self, tokens: Dict[str, torch.Tensor]
    ) -> torch.Tensor:
        """        Формула 2: Получение эмбеддингов токенов.        """
        with torch.no_grad():
            outputs = self.embedding_model(
                input_ids=tokens['input_ids'],
                attention_mask=tokens['attention_mask'])
        return outputs.last_hidden_state
    def extract_syntax_features(
        self, embeddings: torch.Tensor
    ) -> torch.Tensor:
        """        Формула 3: Синтаксические признаки (CNN_syn).        """
        return embeddings
    def fit_lda(self, corpus: List[str]) -> None:
        """        Обучение LDA на корпусе обучающей выборки (формула 4).        """
        if not corpus:
            return
        self._lda_corpus = list(corpus)
        tfidf = self.lda_vectorizer.fit_transform(self._lda_corpus)
        self.lda_model.fit(tfidf)
    def extract_semantic_features(self, text: str) -> np.ndarray:
        """        Формула 4: Семантические признаки через LDA.        """
        if self.lda_model is None or self.lda_vectorizer is None:
            raise FeatureExtractionError("LDA-модель не инициализирована")
        if not hasattr(self.lda_model, 'components_'):
            raise FeatureExtractionError(
                "LDA-модель не обучена. Вызовите fit_lda(corpus).")
        tfidf = self.lda_vectorizer.transform([text])
        topic_dist = self.lda_model.transform(tfidf)[0]
        return topic_dist.astype(np.float32)
    def extract_lexical_mood(self, tokens: List[str]) -> float:
        """        Формула 5: Средняя лексическая тональность.        """
        if not tokens:
            return 0.0
        special = {'[CLS]', '[SEP]', '[PAD]', '[UNK]', '[MASK]'}
        scores = []
        for t in tokens:
```

```
            if t in special:
                continue
            w = t.lower().strip()
            if not w or w in self.stop_words:
                continue
            if w in self.positive_words:
                scores.append(1.0)
            elif w in self.negative_words:
                scores.append(-1.0)
            else:
                scores.append(0.0)
        if not scores:
            return 0.0
        return float(np.mean(scores))
    def extract_burrows_delta(self, tokens: List[str]) -> float:
        """Формула 7: Дельта Берроуза."""
        if not self.token_stats_builder.is_fitted:
            if self.token_stats_builder.document_count == 0:
                raise FeatureExtractionError(
                    "Статистики токенов не обучены. "
                    "Вызовите add_document() и fit().")
            self.token_stats_builder.fit()
        stats = self.token_stats_builder.get_stats()
        mu = stats.get('mu')
        sigma = stats.get('sigma')
        if mu is None or sigma is None:
            return 0.0
        freqs = Counter(tokens)
        total = len(tokens)
        if total == 0:
            return 0.0
        mu_map = self.token_stats_builder.mu_map
        sigma_map = self.token_stats_builder.sigma_map
        present = [t for t in freqs if t in mu_map]
        if not present:
            return 0.0
        eps = self.config.get('epsilon', 1e-8)
        delta_sum = 0.0
        for t in present:
            f_i = freqs[t] / total
            mu_i = mu_map[t]
            sigma_i = max(sigma_map[t], eps)
            delta_sum += abs(f_i - mu_i) / sigma_i
        return float(delta_sum / len(present))
    def extract_parts_distance(
        self, embeddings: torch.Tensor
    ) -> float:
        """Формула 9: Косинусное расстояние между началом и концом текста."""
        if embeddings.dim() == 3:
            batch_size, seq_len, dim = embeddings.shape
        else:
            seq_len, dim = embeddings.shape
```

```
        embeddings = embeddings.unsqueeze(0)
    k = max(1, seq_len // 3)
    emb_first = embeddings[:, :k, :].mean(dim=1)
    emb_last = embeddings[:, -k:, :].mean(dim=1)
    cos_sim = F.cosine_similarity(emb_first, emb_last, dim=1)
    d_parts = torch.clamp(1.0 - cos_sim, 0.0, 2.0)
    return float(d_parts.mean().item())
def _extract_noun_phrases(
    self, tokens: List[str]
) -> Tuple[List[str], List[str]]:
    """    Формула 10: Выделение именных групп (NP) и контекстных слов.    """
    pos_tags = nltk.pos_tag(tokens)
    np_words: List[str] = []
    context_words: List[str] = []
    for word, tag in pos_tags:
        if tag.startswith('NN'):
            np_words.append(word)
        else:
            context_words.append(word)
    return np_words, context_words
def _sentiment_of_words(self, words: List[str]) -> float:
    """        Формулы 11-12: Средняя тональность набора слов.        """
    if not words:
        return 0.0
    pos = sum(1 for w in words if w.lower() in self.positive_words)
    neg = sum(1 for w in words if w.lower() in self.negative_words)
    if pos + neg == 0:
        return 0.0
    return (pos - neg) / (pos + neg)
def extract_semantic_incongruency(
    self, text: str, tokens: List[str]
) -> float:
    """        Формулы 10-13: Индекс семантической инконгруэнтности.        """
    np_words, context_words = self._extract_noun_phrases(tokens)
    sent_topic = self._sentiment_of_words(np_words)
    sent_context = self._sentiment_of_words(context_words)
    return float(abs(sent_topic - sent_context))
def extract_emotional_intensity(self, tokens: List[str]) -> float:
    """        Формула 14: Коэффициент эмоциональной интенсивности.        """
    if len(tokens) < 2:
        return 0.0
    emo_scores = np.array([
        self.emotion_lexicon.get(t.lower(), 0.0) for t in tokens
    ], dtype=np.float32)
    diffs = np.abs(np.diff(emo_scores))
    tau_e = self.config.get('tau_e', 0.5)
    valid = diffs[diffs > tau_e]
    if valid.size == 0:
        return 0.0
    return float(valid.mean())
def extract_hyperbolization(self, tokens: List[str]) -> float:
    """        Формула 15: Индекс гиперболизации.        """
```

```
    if not tokens:
        return 0.0
    weights = np.array([
        self.hyperbole_lexicon.get(t.lower(), 0.0) for t in tokens
    ], dtype=np.float32)
    valid = weights[weights > 0]
    if valid.size == 0:
        return 0.0
    return float(np.clip(valid.mean(), 0.0, 1.0))
def compute_syn_base_profile(
    self, embeddings_list: List[torch.Tensor]
) -> None:
    """Вычисление эталонного синтаксического профиля по обучающей выборке."""
    if not embeddings_list:
        return
    h_syn = self.config.get('hidden_dim', 256)
    profiles = []
    for e in embeddings_list:
        h = e.mean(dim=1).squeeze(0)
        if h.shape[0] >= h_syn:
            profiles.append(h[:h_syn])
        else:
            profiles.append(F.pad(h, (0, h_syn - h.shape[0])))
    self.syn_base_profile = torch.stack(profiles).mean(dim=0)
def extract_stylistic_dissonance(
    self, embeddings: torch.Tensor
) -> float:
    """Формула 16: Коэффициент стилистического диссонанса. """
    h_syn = self.config.get('hidden_dim', 256)
    if self.syn_base_profile is None:
        raise FeatureExtractionError(
            "Эталонный синтаксический профиль не вычислен. "
            "Вызовите compute_syn_base_profile().")
    h_emb = embeddings.mean(dim=1).squeeze(0) \
        if embeddings.dim() == 3 else embeddings.mean(dim=0)
    if h_emb.shape[0] >= h_syn:
        h_syn_vec = h_emb[:h_syn]
    else:
        h_syn_vec = F.pad(h_emb, (0, h_syn - h_emb.shape[0]))
    base = self.syn_base_profile.to(h_syn_vec.device)
    eps = self.config.get('epsilon', 1e-8)
    diff = torch.norm(h_syn_vec - base)
    norm_base = torch.norm(base)
    return float(diff / (norm_base + eps))
def extract_contextual_dissonance(
    self,
    current_embeddings: torch.Tensor,
    context_embeddings: Optional[torch.Tensor] = None
) -> float:
    """Формула 20: Коэффициент контекстного диссонанса. """
    sp_current = current_embeddings.mean(dim=1).squeeze(0) \
        if current_embeddings.dim() == 3 \
```

```
        else current_embeddings.mean(dim=0)
    if context_embeddings is None:
        return 0.0
    sp_context = context_embeddings.mean(dim=1).squeeze(0) \
        if context_embeddings.dim() == 3 \
        else context_embeddings.mean(dim=0)
    cos_sim = F.cosine_similarity(
        sp_current.unsqueeze(0), sp_context.unsqueeze(0))
    return float(torch.clamp(1.0 - cos_sim, 0.0, 2.0))
def extract_all_features(
    self,
    text: str,
    tokens: Dict[str, torch.Tensor],
    embeddings: torch.Tensor,
    context_embeddings: Optional[torch.Tensor] = None
) -> Dict[str, Any]:
    """Извлечение всех признаков для модели."""
    token_list = tokens['tokens']
    semantic_features = self.extract_semantic_features(text)
    semantic_tensor = torch.tensor(
        semantic_features, dtype=torch.float32)
    lexical_mood = self.extract_lexical_mood(token_list)
    burrows_delta = self.extract_burrows_delta(token_list)
    parts_distance = self.extract_parts_distance(embeddings)
    semantic_incongruency = self.extract_semantic_incongruency(
        text, token_list)
    emotional_intensity = self.extract_emotional_intensity(token_list)
    hyperbolization = self.extract_hyperbolization(token_list)
    stylistic_dissonance = self.extract_stylistic_dissonance(embeddings)
    contextual_dissonance = self.extract_contextual_dissonance(
        embeddings, context_embeddings)
    features = {
        'input_ids': tokens['input_ids'],
        'attention_mask': tokens['attention_mask'],
        'embeddings': embeddings,
        'syntax_features': self.extract_syntax_features(embeddings),
        'semantic_features': semantic_tensor,
        'lexical_mood': torch.tensor(
            [lexical_mood], dtype=torch.float32),
        'burrows_delta': torch.tensor(
            [burrows_delta], dtype=torch.float32),
        'parts_distance': torch.tensor(
            [parts_distance], dtype=torch.float32),
        'semantic_incongruency': torch.tensor(
            [semantic_incongruency], dtype=torch.float32),
        'emotional_intensity': torch.tensor(
            [emotional_intensity], dtype=torch.float32),
        'hyperbolization': torch.tensor(
            [hyperbolization], dtype=torch.float32),
        'stylistic_dissonance': torch.tensor(
            [stylistic_dissonance], dtype=torch.float32),
        'contextual_dissonance': torch.tensor(
```

```
            [contextual_dissonance], dtype=torch.float32),
        'token_stats': self.token_stats_builder.get_stats()}
    return features
  def _clean_text(self, text: str) -> str:
    """
    Очистка входного текста.
    Args:
      text: Входной текст
    Returns:
      Очищенный текст
    """
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'[^\w\s.,!?\'"()-]', '', text)
    return text.strip()
  def _compute_sentiment(self, words: List[str]) -> float:
    """
    Вычисление тональности для списка слов.
    Args:
      words: Список слов
    Returns:
      Оценка тональности в диапазоне [-1, 1]
    """
    return self._sentiment_of_words(words)
```

src/hybrid_qualifier/utils/token_stats_builder.py

```
"""
Построитель статистик токенов для вычисления дельты Берроуза.
Реализует формулы 7-8: Δ_B = (1/|V_pr|) * Σ |f_i - μ_i| / σ_i.
Статистики μ_i, σ_i вычисляются на обучающей выборке С3.
Единицы частот - относительные (доля от общего числа токенов документа).
"""
from typing import Dict, List, Optional, Any
from collections import Counter
import numpy as np
import json
import os
class TokenStatsBuilder:
  def __init__(self, max_vocab_size: int = 100000):
    """ Инициализация построителя статистик. """
    self.max_vocab_size = max_vocab_size
    self.token_to_idx: Dict[str, int] = {}
    self.idx_to_token: Dict[int, str] = {}
    self.token_frequencies: np.ndarray = np.zeros(
      max_vocab_size, dtype=np.float64)
    self.document_frequencies: np.ndarray = np.zeros(
      max_vocab_size, dtype=np.int64)
    # Относительные частоты по документам для вычисления σ
    self._doc_relative_freqs: Dict[int, List[float]] = {}
    self.total_tokens: int = 0
    self.document_count: int = 0
    self.vocab_size: int = 0
    # Вычисленные статистики (относительные частоты)
```

```
    self.mu: Optional[np.ndarray] = None
    self.sigma: Optional[np.ndarray] = None
    self.mu_map: Dict[str, float] = {}
    self.sigma_map: Dict[str, float] = {}
    self.eps: float = 1e-8
    self.is_fitted: bool = False
def add_document(self, tokens: List[str]) -> None:
    """ Добавление документа в статистику. """
    if not tokens:
        return
    doc_counter = Counter(tokens)
    total = len(tokens)
    self.document_count += 1
    self.total_tokens += total
    for token, count in doc_counter.items():
        idx = self._get_token_idx(token)
        self.token_frequencies[idx] += count
        self.document_frequencies[idx] += 1
        rel_freq = count / total
        self._doc_relative_freqs.setdefault(idx, []).append(rel_freq)
def _get_token_idx(self, token: str) -> int:
    """Получение или создание индекса токена."""
    if token not in self.token_to_idx:
        if self.vocab_size >= self.max_vocab_size:
            self._prune_vocabulary()
        idx = self.vocab_size
        self.token_to_idx[token] = idx
        self.idx_to_token[idx] = token
        self.vocab_size += 1
    return self.token_to_idx[token]
def _prune_vocabulary(self) -> None:
    """Сокращение словаря до max_vocab_size."""
    freq_idx = [
        (self.token_frequencies[i], i)
        for i in range(self.vocab_size)]
    freq_idx.sort(reverse=True)
    keep_count = self.max_vocab_size // 2
    keep_indices = {idx for _, idx in freq_idx[:keep_count]}
    new_token_to_idx: Dict[str, int] = {}
    new_idx_to_token: Dict[int, str] = {}
    new_freq = np.zeros(self.max_vocab_size, dtype=np.float64)
    new_doc_freq = np.zeros(self.max_vocab_size, dtype=np.int64)
    new_rel: Dict[int, List[float]] = {}
    new_idx = 0
    for token, old_idx in self.token_to_idx.items():
        if old_idx in keep_indices:
            new_token_to_idx[token] = new_idx
            new_idx_to_token[new_idx] = token
            new_freq[new_idx] = self.token_frequencies[old_idx]
            new_doc_freq[new_idx] = self.document_frequencies[old_idx]
            if old_idx in self._doc_relative_freqs:
                new_rel[new_idx] = self._doc_relative_freqs[old_idx]
```

```
                new_idx += 1
            self.token_to_idx = new_token_to_idx
            self.idx_to_token = new_idx_to_token
            self.token_frequencies = new_freq
            self.document_frequencies = new_doc_freq
            self._doc_relative_freqs = new_rel
            self.vocab_size = new_idx
        def fit(self) -> None:
            """ Вычисление μ и σ относительных частот токенов по обучающей выборке (формула 7). """
            if self.document_count == 0:
                raise ValueError(
                    "Нет добавленных документов. "
                    "Сначала вызовите add_document().")
            self.mu = np.zeros(self.vocab_size, dtype=np.float64)
            self.sigma = np.zeros(self.vocab_size, dtype=np.float64)
            for idx in range(self.vocab_size):
                rel_freqs = self._doc_relative_freqs.get(idx, [])
                if rel_freqs:
                    self.mu[idx] = float(np.mean(rel_freqs))
                    self.sigma[idx] = float(np.std(rel_freqs))
                else:
                    self.mu[idx] = 0.0
                    self.sigma[idx] = self.eps
            self.sigma = np.maximum(self.sigma, self.eps)
            self.is_fitted = True
            self.mu_map = {
                self.idx_to_token[i]: float(self.mu[i])
                for i in range(self.vocab_size)}
            self.sigma_map = {
                self.idx_to_token[i]: float(self.sigma[i])
                for i in range(self.vocab_size)}
        def compute_delta(self, tokens: List[str]) -> float:
            """
            Формула 7: дельта Берроуза.
            Δ_B = (1/|V_pr|) * Σ |f_i - μ_i| / σ_i
            Args:
                tokens: Список токенов из документа
            Returns:
                Дельта Берроуза
            """
            if not self.is_fitted:
                self.fit()
            if not tokens:
                return 0.0
            freqs = Counter(tokens)
            total = len(tokens)
            present = [t for t in freqs if t in self.mu_map]
            if not present:
                return 0.0
            delta_sum = 0.0
            for t in present:
```

```
            f_i = freqs[t] / total
            mu_i = self.mu_map[t]
            sigma_i = max(self.sigma_map[t], self.eps)
            delta_sum += abs(f_i - mu_i) / sigma_i
        return float(delta_sum / len(present))
    def get_stats(self) -> Dict[str, Any]:
        """ Получение статистик в виде массивов для передачи в модель. """
        if not self.is_fitted:
            self.fit()
        return {
            'mu': self.mu[:self.vocab_size]
            if self.mu is not None else None,
            'sigma': self.sigma[:self.vocab_size]
            if self.sigma is not None else None,
            'vocab_size': self.vocab_size}
    def save_stats(self, filepath: str) -> None:
        """Сохранение статистик в файл."""
        if not self.is_fitted:
            self.fit()
        os.makedirs(os.path.dirname(filepath), exist_ok=True)
        stats = {
            'tokens': list(self.token_to_idx.keys()),
            'mu': [float(self.mu[i]) for i in range(self.vocab_size)],
            'sigma': [float(self.sigma[i]) for i in range(self.vocab_size)],
            'document_count': self.document_count,
            'total_tokens': self.total_tokens,
            'vocab_size': self.vocab_size}
        with open(filepath, 'w', encoding='utf-8') as f:
            json.dump(stats, f, indent=2)
    def load_stats(self, filepath: str) -> None:
        """Загрузка статистик из файла."""
        with open(filepath, 'r', encoding='utf-8') as f:
            stats = json.load(f)
        tokens = stats['tokens']
        vocab_size = len(tokens)
        self.token_to_idx = {t: i for i, t in enumerate(tokens)}
        self.idx_to_token = {i: t for i, t in enumerate(tokens)}
        self.mu = np.array(stats['mu'], dtype=np.float64)
        self.sigma = np.array(stats['sigma'], dtype=np.float64)
        self.vocab_size = vocab_size
        self.document_count = stats['document_count']
        self.total_tokens = stats['total_tokens']
        self.is_fitted = True
        self.mu_map = {
            self.idx_to_token[i]: float(self.mu[i])
            for i in range(vocab_size)}
        self.sigma_map = {
            self.idx_to_token[i]: float(self.sigma[i])
            for i in range(vocab_size)}
    def clear(self) -> None:
        """Очистка всех статистик."""
        self.token_to_idx.clear()
```

```
self.idx_to_token.clear()
self.token_frequencies.fill(0)
self.document_frequencies.fill(0)
self._doc_relative_freqs.clear()
self.total_tokens  = 0
self.document_count  = 0
self.vocab_size  = 0
self.mu  = None
self.sigma  = None
self.mu_map  = {}
self.sigma_map  = {}
self.is_fitted  = False
```

ПРИЛОЖЕНИЕ Д

Специфицированный протокол взаимодействия решения с платформами сбора данных - HQP. Элемент adapter.py

Представленный ниже протокол относится к классу семантических интеграционных протоколов прикладного уровня, поскольку обеспечивает взаимодействие между парсинговой системой и библиотекой квалификации. Реализация метода квалификации в форме библиотеки не упраздняет, а, напротив, повышает значимость разработки данного протокола: библиотека выступает в качестве единой точки входа для всех операций квалификации в составе парсинговой системы, что требует формализованного и семантически согласованного взаимодействия на уровне прикладного интерфейса

Структура протокола

```
hqp/
├── __init__.py            # Публичный интерфейс
├── adapter.py             # Адаптер с исправленными методами
                Единственная точка интеграции между протоколом и библиотекой
                Преобразует все форматы данных
                Управляет production-компонентами (Circuit Breaker, Cache, Metrics)
                Обеспечивает надежность (таймауты, fallback, thread safety)
                Содержит основную логику протокола
├── health.py              # Health checks
├── requirements.txt       # Зависимости
├── protocol/
│   ├── __init__.py
│   ├── types.py           # Типы данных
│   ├── validation.py      # УРОВЕНЬ 1: Валидация
│   ├── enrichment.py      # УРОВЕНЬ 2: Обогащение
│   ├── canonical.py       # УРОВЕНЬ 3: Каноническая трансформация
│   ├── qualification.py   # УРОВЕНЬ 4: Квалификация (ядро)
│   ├── formatting.py      # УРОВЕНЬ 5: Форматирование
│   └── errors.py          # Ошибки
├── core/
│   ├── __init__.py
│   ├── threading.py       # Потокобезопасность
│   ├── circuit_breaker.py # Circuit Breaker
│   ├── cache.py           # Кэширование
│   └── metrics.py         # Метрики
├── utils/
│   ├── __init__.py
│   ├── language.py        # Определение языка
│   └── entities.py        # Извлечение сущностей
└── config/
    └── default.py         # Конфигурация
```

hqp/adapter.py

```
from typing import Dict, List, Optional, Any
import time
import uuid
import logging
import concurrent.futures
from datetime import datetime
from .protocol.types import (
    Request, Response, QualifierResult, CategoryResult, SetResult,
    SeverityResult, SentimentResult, SeverityLevel, SentimentLabel,
    Status, MetaInfo, ErrorInfo, Text, Context, SourcePlatform, Priority)
from .protocol.validation import validate_request
from .protocol.enrichment import enrich_text
from .protocol.canonical import to_canonical
from .protocol.errors import create_error, ProtocolError
from .config.default import Config
from .core.threading import ThreadSafeAdapter, AsyncThreadPool
from .core.circuit_breaker import CircuitBreaker, CircuitBreakerConfig, CircuitBreakerOpenError
from .core.cache import RequestCache
from .core.metrics import MetricsCollector, Timer
logger = logging.getLogger(__name__)
# Импорт библиотеки hybrid-qualifier
try:
    from hybrid_qualifier import (
        HybridQualifier as LibHybridQualifier,
        __version__ as LIB_VERSION    )
    from hybrid_qualifier.models.pretrained import load_pretrained_model
    LIBRARY_AVAILABLE = True
except ImportError as e:
    LIBRARY_AVAILABLE = False
    LibHybridQualifier = None
    load_pretrained_model = None
    LIB_VERSION = None
    logger.error(f"Библиотека hybrid-qualifier не найдена: {e}")
class HQPAdapter:
    """
    Адаптер, преобразующий HQP протокол в вызовы библиотеки hybrid-qualifier.
    """
    # Мапинг категорий (строка -> ID)
    CATEGORY_MAP = {
        'unknown': 0,
        'technical': 1,
        'sales': 2,
        'support': 3,
        'complaint': 4,
        'feedback': 5,
        'inquiry': 6,
        'other': 7    }
    # Мапинг ID категорий в строки
    CATEGORY_NAME_MAP = {v: k for k, v in CATEGORY_MAP.items()}
    # Названия уровней тяжести
    SEVERITY_NAMES = {
```

```
    0: 'noise',
    1: 'light',
    2: 'medium',
    3: 'medium_heavy',
    4: 'heavy'    }
# Мапинг тональности
SENTIMENT_MAP = {
    'negative': SentimentLabel.NEGATIVE,
    'neutral': SentimentLabel.NEUTRAL,
    'positive': SentimentLabel.POSITIVE    }
# Порядок меток тональности в выходе библиотеки
SENTIMENT_ORDER = ['negative', 'neutral', 'positive']
def __init__(self, config: Optional[Config] = None):
    """
    Инициализация адаптера.
    Args:
        config: Конфигурация
    """
    self.config = config or Config()
    self._library_qualifier: Optional[LibHybridQualifier] = None
    self._initialization_error: Optional[Exception] = None
    self._shutdown_called = False
    # Production компоненты
    self._circuit_breaker = CircuitBreaker(
        CircuitBreakerConfig(
            failure_threshold=self.config.circuit_breaker_failure_threshold,
            recovery_timeout=self.config.circuit_breaker_recovery_timeout            )        )
    self._cache = RequestCache(
        max_size=self.config.cache_size,
        ttl=self.config.cache_ttl        )
    self._metrics = MetricsCollector()
    self._thread_safe = ThreadSafeAdapter(self, self.config.max_concurrent_calls)
    self._async_pool = AsyncThreadPool(max_workers=self.config.max_workers)
    # Инициализация библиотеки
    try:
        self._initialize_library()
    except Exception as e:
        self._initialization_error = e
        logger.critical(f"Failed to initialize library: {e}")
        if not self.config.fallback_enabled:
            raise
def _build_library_config(self) -> Dict[str, Any]:
    """
    Формирование конфигурации для библиотеки hybrid-qualifier.
    Включает архитектурные параметры, которых нет в Config.to_dict().
    """
    cfg = self.config.to_dict()
    cfg.update({
        'embedding_dim': getattr(self.config, 'embedding_dim', 768),
        'hidden_dim': getattr(self.config, 'hidden_dim', 256),
        'num_categories': getattr(self.config, 'num_categories', 10),
        'num_severity_levels': getattr(self.config, 'num_severity_levels', 5),
```

```
        'num_sentiment_classes': getattr(self.config, 'num_sentiment_classes', 3),
        'vocab_size': getattr(self.config, 'vocab_size', 30522),
        'num_topics': getattr(self.config, 'num_topics', 5),
        'mood_dim': getattr(self.config, 'mood_dim', 6),
        'max_seq_length': getattr(self.config, 'max_seq_length', 512),
        'attention_heads': getattr(self.config, 'attention_heads', 8),
        'gamma': getattr(self.config, 'gamma', 1.0),
        'tau_e': getattr(self.config, 'tau_e', 0.5),
        'lambda_reg': getattr(self.config, 'lambda_reg', 0.1),
        'epsilon': getattr(self.config, 'epsilon', 1e-8),
        'short_text_threshold': getattr(self.config, 'short_text_threshold', 20),
        'short_text_tau': getattr(self.config, 'short_text_tau', 10),
        'max_text_length': self.config.max_text_length,
    })
    return cfg
def _initialize_library(self):
    """Инициализация библиотеки hybrid-qualifier."""
    if not LIBRARY_AVAILABLE:
        raise create_error(
            "INT_001",
            details={"message": "Библиотека hybrid-qualifier не установлена"}            )
    try:
        with Timer(self._metrics, 'library_init'):
            self._library_qualifier = LibHybridQualifier(
                model_path=self.config.model_path,
                device=self.config.device,
                config=self._build_library_config()                )
            self._library_qualifier.load_model()
            # Калибровка, требуемая библиотекой
            self._calibrate_library()
        logger.info("Библиотека hybrid-qualifier инициализирована")
    except Exception as e:
        logger.error(f"Ошибка инициализации библиотеки: {e}")
        raise create_error(
            "INT_003",
            details={"error": str(e), "message": "Модель не загружена"}            )
def _calibrate_library(self):
    """
    Калибровка внутренних компонентов библиотеки:
    - обучение LDA на корпусе (если доступен),
    - обучение статистик токенов (если доступен корпус),
    - установка эталонных профилей.
    """
    qualifier = self._library_qualifier
    if qualifier is None:
        return
    corpus = getattr(self.config, 'calibration_corpus', None)
    if corpus:
        # Обучение LDA
        try:
            qualifier.fit_lda(corpus)
        except Exception as e:
```

```
            logger.warning(f"fit_lda failed: {e}")
        # Обучение статистик токенов
        try:
            tokenized_corpus = []
            if qualifier.feature_extractor is not None:
                for doc in corpus:
                    tokenized_corpus.append(
                        qualifier.feature_extractor.tokenize(doc)['tokens']
                    )
            if tokenized_corpus:
                qualifier.fit_token_stats(tokenized_corpus)
        except Exception as e:
            logger.warning(f"fit_token_stats failed: {e}")
    # Установка эталонных профилей, если они переданы в конфиге
    calibration = getattr(self.config, 'calibration_data', None)
    if calibration and qualifier.model is not None:
        try:
            import torch
            qualifier.model.set_calibration(
                mu_priz=torch.tensor(calibration['mu_priz'], dtype=torch.float32),
                sigma_priz=torch.tensor(calibration['sigma_priz'], dtype=torch.float32),
                mu_sp=torch.tensor(calibration['mu_sp'], dtype=torch.float32),
                sigma_sp=torch.tensor(calibration['sigma_sp'], dtype=torch.float32),
                sp_base=torch.tensor(calibration['sp_base'], dtype=torch.float32),
                syn_base_profile=torch.tensor(calibration['syn_base_profile'], dtype=torch.float32),        )
        except Exception as e:
            logger.warning(f"set_calibration failed: {e}")
def _is_library_available(self) -> bool:
    """Проверка доступности библиотеки."""
    if self._library_qualifier is None:
        return False
    if self._initialization_error is not None:
        return False
    return True
def _get_fallback_response(self, request_id: str, error_msg: str) -> Dict:
    """Получение fallback ответа при недоступности библиотеки."""
    if self.config.fallback_enabled:
        return {
            "id": request_id,
            "status": "error",
            "meta": {
                "timestamp": datetime.utcnow().isoformat() + "Z",
                "processing_time_ms": 0,
                "version": "1.0",
                "warnings": []             },
            "error": {
                "code": "CKT_001",
                "message": f"Service unavailable: {error_msg}",
                "recoverable": True,
                "details": {"fallback": True}             }         }
    return Response.error(
        request_id=request_id,
```

```
        error=ErrorInfo(
            code="CKT_001",
            message=f"Service unavailable: {error_msg}",
            recoverable=True
        ),
        processing_time_ms=0
    ).to_dict()
def _fallback_for_circuit_breaker(
    self,
    text: str,
    context: Optional[Dict] = None,
    options: Optional[Dict] = None
) -> Dict:
    """Обёртка fallback с сигнатурой, совместимой с CircuitBreaker.call."""
    return self._get_fallback_response(
        str(uuid.uuid4()),
        "Circuit breaker open or library failure"        )
def qualify(
    self,
    text: str,
    context: Optional[Dict] = None,
    options: Optional[Dict] = None
) -> Dict:
    """
    Полная квалификация через протокол HQP.
    Args:
        text: Текст для анализа
        context: Контекст
        options: Опции
    Returns:
        Результат квалификации в формате HQP
    """
    # Проверка доступности библиотеки
    if not self._is_library_available():
        if self.config.fallback_enabled:
            logger.warning("Library not available, using fallback")
            return self._get_fallback_response(
                str(uuid.uuid4()), "Library not initialized"                )
        raise self._initialization_error or RuntimeError("Library not available")
    # Проверка кэша
    cache_result = self._cache.get(text, context)
    if cache_result:
        self._metrics.counter('cache_hit')
        logger.debug(f"Cache hit for text: {text[:50]}")
        return cache_result
    self._metrics.counter('cache_miss')
    # Вызов через Circuit Breaker
    try:
        result = self._circuit_breaker.call(
            self._do_qualify,
            self._fallback_for_circuit_breaker,
            text, context, options            )
```

```
        if result.get('status') == 'success':
            self._cache.set(text, context, result)
        return result
    except CircuitBreakerOpenError as e:
        self._metrics.counter('circuit_breaker_open')
        logger.error(f"Circuit breaker open: {e}")
        return self._get_fallback_response(str(uuid.uuid4()), str(e))
    except Exception as e:
        self._metrics.counter('qualify_error')
        logger.error(f"Qualify error: {e}", exc_info=True)
        return Response.error(
            request_id=str(uuid.uuid4()),
            error=ErrorInfo(
                code="PRD_001",
                message=f"Production error: {str(e)}",
                recoverable=True ),
            processing_time_ms=0
        ).to_dict()
def _do_qualify(
    self,
    text: str,
    context: Optional[Dict] = None,
    options: Optional[Dict] = None
) -> Dict:
    start_time = time.time()
    request_id = str(uuid.uuid4())
    try:
        self._metrics.counter('qualify_request')
        with Timer(self._metrics, 'qualify_duration'):
            # 1. Формирование запроса
            request = self._create_request(text, context, options)
            request_id = request.id
            # 2. Уровни протокола
            validated = validate_request(request)
            enriched = enrich_text(validated)
            canonical = to_canonical(enriched)
            # 3. Вызов библиотеки
            options = options or {}
            include_features = options.get('return_features', False)
            include_embeddings = options.get('return_embeddings', False)
            context_texts = options.get('context_texts')
            lib_result = self._call_library_with_timeout(
                canonical.text.normalized,
                include_features=include_features,
                include_embeddings=include_embeddings,
                context_texts=context_texts )
          # Проверка на ошибку библиотеки
            if isinstance(lib_result, dict) and lib_result.get('success') is False:
                raise RuntimeError(
                    lib_result.get('error', 'Library reported failure') )
            # 4. Преобразование результата
            hqp_result = self._convert_library_result_to_hqp(
```

```
            lib_result,
            request_id,
            return_probs=options.get('return_probs', False)
        )
        # 5. Форматирование ответа
        processing_time_ms = (time.time() - start_time) * 1000
        if hqp_result.metadata is None:
            hqp_result.metadata = {}
        hqp_result.metadata['processing_time_ms'] = processing_time_ms
        response = Response.success(
            request_id=request_id,
            result=hqp_result,
            processing_time_ms=processing_time_ms
        )
        self._metrics.counter('qualify_success')
        self._metrics.gauge('confidence', hqp_result.confidence)
        self._metrics.gauge('anomaly', hqp_result.anomaly)
        self._metrics.record_circuit_breaker_state(
            self._circuit_breaker.state.value
        )
        return response.to_dict()
    except TimeoutError as e:
        processing_time_ms = (time.time() - start_time) * 1000
        self._metrics.counter('qualify_timeout')
        logger.error(f"Timeout: {e}")
        error_info = ErrorInfo(
            code="TO_001",
            message=f"Timeout: {str(e)}",
            recoverable=True,
            details={"timeout_seconds": self.config.library_timeout_seconds}
        )
        response = Response.error(
            request_id=request_id,
            error=error_info,
            processing_time_ms=processing_time_ms
        )
        return response.to_dict()
    except Exception as e:
        processing_time_ms = (time.time() - start_time) * 1000
        self._metrics.counter('qualify_exception')
        if isinstance(e, ProtocolError):
            error_info = ErrorInfo(
                code=e.code,
                message=e.message,
                recoverable=e.recoverable,
                details=e.details
            )
        else:
            error_info = ErrorInfo(
                code="INT_001",
                message=f"Qualification error: {str(e)}",
                recoverable=False,
                details={"error_type": type(e).__name__}
            )
        response = Response.error(
            request_id=request_id,
            error=error_info,
            processing_time_ms=processing_time_ms
        )
        return response.to_dict()
```

```
def _call_library_with_timeout(
    self,
    text: str,
    include_features: bool = False,
    include_embeddings: bool = False,
    context_texts: Optional[List[str]] = None
) -> Dict:
    """
    Вызов библиотеки с таймаутом.
    Args:
        text: Текст для анализа
        include_features: Включать признаки
        include_embeddings: Включать эмбеддинги
        context_texts: Предыдущие сообщения диалога
    Returns:
        Результат от библиотеки
    """
    if not self._is_library_available():
        raise RuntimeError("Library not available")
    with concurrent.futures.ThreadPoolExecutor(max_workers=1) as executor:
        future = executor.submit(
            self._thread_safe.call,
            self._library_qualifier.analyze,
            text=text,
            include_features=include_features,
            include_embeddings=include_embeddings,
            context_texts=context_texts )
        try:
            return future.result(timeout=self.config.library_timeout_seconds)
        except concurrent.futures.TimeoutError:
            raise TimeoutError(
                f'Library call timeout after '
                f'{self.config.library_timeout_seconds}s" )
def _convert_library_result_to_hqp(
    self,
    lib_result: Dict[str, Any],
    request_id: str,
    return_probs: bool = False
) -> QualifierResult:
    """
    Преобразование результата библиотеки hybrid-qualifier в формат HQP.
    Args:
        lib_result: Результат от библиотеки
        request_id: ID запроса
        return_probs: Возвращать вероятности
    Returns:
        QualifierResult
    """
    # 1. Категория
    raw_category = lib_result.get('category', 0)
    if isinstance(raw_category, str):
        category_name = raw_category
```

```
    category_id = self.CATEGORY_MAP.get(category_name, 0)
else:
    category_id = int(raw_category)
    category_name = self.CATEGORY_NAME_MAP.get(category_id, 'unknown')
category_confidence = float(lib_result.get('category_prob', 0.0))
category = CategoryResult(
    id=category_id,
    name=category_name,
    confidence=category_confidence        )
# 2. Set
raw_set = lib_result.get('set') or {}
set_result = SetResult(
    id=int(raw_set.get('id', 0)),
    name=raw_set.get('name', 'unknown'),
    confidence=float(raw_set.get('confidence', 0.0))       )
# 3. Severity
severity_level = int(lib_result.get('severity', 0))
severity_probs_raw = lib_result.get('severity_probs')
severity_probs: Optional[Dict[str, float]] = None
if return_probs and severity_probs_raw:
    severity_probs = {
        self.SEVERITY_NAMES.get(i, str(i)): float(p)
        for i, p in enumerate(severity_probs_raw)            }
severity_confidence = (
    severity_probs_raw[severity_level]
    if severity_probs_raw and 0 <= severity_level < len(severity_probs_raw)
    else 0.5       )
severity = SeverityResult(
    level=SeverityLevel.from_int(severity_level),
    name=self.SEVERITY_NAMES.get(severity_level, 'unknown'),
    confidence=float(severity_confidence),
    probabilities=severity_probs        )
# 4. Sentiment
sentiment_label_str = lib_result.get('sentiment', 'neutral')
sentiment_label = self.SENTIMENT_MAP.get(
    sentiment_label_str, SentimentLabel.NEUTRAL        )
sentiment_probs_raw = lib_result.get('sentiment_probs')
sentiment_probs: Optional[Dict[str, float]] = None
if return_probs and sentiment_probs_raw:
    sentiment_probs = {
        label: float(p)
        for label, p in zip(self.SENTIMENT_ORDER, sentiment_probs_raw)            }
if sentiment_probs_raw:
    try:
        idx = self.SENTIMENT_ORDER.index(sentiment_label_str)
        sentiment_confidence = float(sentiment_probs_raw[idx])
    except (ValueError, IndexError):
        sentiment_confidence = float(max(sentiment_probs_raw))
else:
    sentiment_confidence = 0.5
sentiment = SentimentResult(
    label=sentiment_label,
```

```
        confidence=sentiment_confidence,
        probabilities=sentiment_probs        )
    # 5. Аномальность (proxy - sarcasm_prob)
    anomaly = float(lib_result.get('sarcasm_prob', 0.0))
    # 6. Общая уверенность - используем category_prob
    confidence = category_confidence
    # 7. Признаки
    features = lib_result.get('features')
    # 8. Метаданные
    metadata = {
        'request_id': request_id,
        'model_version': LIB_VERSION or '1.0.0',
        'library_version': LIB_VERSION or '1.0.0',
        'processing_time_ms': float(lib_result.get('processing_time', 0.0)) * 1000.0,
        'token_count': int(lib_result.get('token_count', 0)),
        'sarcasm': bool(lib_result.get('sarcasm', False)),
        'sarcasm_prob': float(lib_result.get('sarcasm_prob', 0.0)),        }
    return QualifierResult(
        category=category,
        set=set_result,
        severity=severity,
        sentiment=sentiment,
        anomaly=anomaly,
        confidence=confidence,
        features=features,
        metadata=metadata        )
def _create_request(
    self,
    text: str,
    context: Optional[Dict],
    options: Optional[Dict]
) -> Request:
    """Создание HQP запроса из простых параметров."""
    text_obj = Text(
        content=text,
        length=len(text)        )
    ctx = Context()
    if context:
        if 'source' in context:
            ctx.source = SourcePlatform(context['source'])
        if 'priority' in context:
            ctx.priority = Priority(context['priority'])
        if 'user_id' in context:
            ctx.user_id = context['user_id']
        if 'session_id' in context:
            ctx.session_id = context['session_id']
        if 'organization_id' in context:
            ctx.organization_id = context['organization_id']
        excluded_keys = {
            'source', 'priority', 'user_id', 'session_id', 'organization_id'            }
        ctx.metadata = {
            k: v for k, v in context.items() if k not in excluded_keys            }
```

```
    return Request(
        id=str(uuid.uuid4()),
        text=text_obj,
        context=ctx,
        options=options or {}        )
async def qualify_async(
    self,
    text: str,
    context: Optional[Dict] = None,
    options: Optional[Dict] = None
) -> Dict:
    """Асинхронная квалификация."""
    return await self._async_pool.run_in_thread(
        self.qualify,
        text, context, options        )
async def qualify_async_with_timeout(
    self,
    text: str,
    context: Optional[Dict] = None,
    options: Optional[Dict] = None,
    timeout: Optional[float] = None
) -> Dict:
    """Асинхронная квалификация с таймаутом."""
    import asyncio
    timeout_sec = timeout or self.config.request_timeout_seconds
    try:
        return await asyncio.wait_for(
            self.qualify_async(text, context, options),
            timeout=timeout_sec            )
    except asyncio.TimeoutError:
        logger.error(f"Async timeout after {timeout_sec}s")
        return Response.error(
            request_id=str(uuid.uuid4()),
            error=ErrorInfo(
                code="TO_001",
                message=f"Async timeout after {timeout_sec}s",
                recoverable=True                ),
            processing_time_ms=timeout_sec * 1000            ).to_dict()
def qualify_batch(
    self,
    texts: List[str],
    contexts: Optional[List[Dict]] = None,
    options: Optional[Dict] = None,
    max_batch_size: int = 32
) -> List[Dict]:
    """
    Пакетная квалификация.
    Args:
        texts: Список текстов
        contexts: Список контекстов
        options: Опции
        max_batch_size: Максимальный размер пакета
```

```
    Returns:
        Список результатов
    """
    results = []
    total = len(texts)
    for i in range(0, total, max_batch_size):
        batch_texts = texts[i:i + max_batch_size]
        batch_contexts = (
            contexts[i:i + max_batch_size] if contexts else None
        )
        with Timer(self._metrics, 'batch_duration',
                   {'size': str(len(batch_texts))}):
            if self._library_qualifier and hasattr(
                self._library_qualifier, 'analyze_batch'
            ):
                batch_results = self._process_batch_library(
                    batch_texts, batch_contexts, options
                )
            else:
                batch_results = self._process_batch_sequential(
                    batch_texts, batch_contexts, options
                )
            results.extend(batch_results)
    # Очистка GPU памяти
    try:
        import torch
        if torch.cuda.is_available():
            torch.cuda.empty_cache()
    except ImportError:
        pass
    return results
def _process_batch_library(
    self,
    texts: List[str],
    contexts: Optional[List[Dict]],
    options: Optional[Dict]
) -> List[Dict]:
    """Пакетная обработка через библиотеку."""
    try:
        options = options or {}
        include_features = options.get('return_features', False)
        if not self._is_library_available():
            raise RuntimeError("Library not available")
        with concurrent.futures.ThreadPoolExecutor(max_workers=1) as executor:
            future = executor.submit(
                self._thread_safe.call,
                self._library_qualifier.analyze_batch,
                texts=texts,
                include_features=include_features
            )
            try:
                batch_results = future.result(
                    timeout=self.config.library_timeout_seconds * 2
                )
            except concurrent.futures.TimeoutError:
                raise TimeoutError("Batch library call timeout")
        responses = []
        for lib_result in batch_results:
```

```
            request_id = str(uuid.uuid4())
            if isinstance(lib_result, dict) and lib_result.get('success') is False:
                responses.append(
                    Response.error(
                        request_id=request_id,
                        error=ErrorInfo(
                            code="PROC_003",
                            message=lib_result.get('error', 'Batch item failed'),
                            recoverable=True
                        ),
                        processing_time_ms=0
                    ).to_dict()
                )
                continue
            hqp_result = self._convert_library_result_to_hqp(
                lib_result,
                request_id,
                return_probs=options.get('return_probs', False)
            )
            response = Response.success(
                request_id=request_id,
                result=hqp_result,
                processing_time_ms=0
            )
            responses.append(response.to_dict())
        return responses
    except Exception as e:
        logger.error(f"Batch library error: {e}, falling back to sequential")
        return self._process_batch_sequential(texts, contexts, options)
def _process_batch_sequential(
    self,
    texts: List[str],
    contexts: Optional[List[Dict]],
    options: Optional[Dict]
) -> List[Dict]:
    """Последовательная обработка (fallback)."""
    results = []
    for i, text in enumerate(texts):
        ctx = contexts[i] if contexts and i < len(contexts) else None
        result = self.qualify(text, ctx, options)
        results.append(result)
    return results
def get_metrics(self) -> Dict[str, Any]:
    """Получение метрик для мониторинга."""
    try:
        cache_stats = self._cache.cache.stats() if hasattr(
            self._cache, 'cache'
        ) else {}
    except Exception:
        cache_stats = {}
    return {
        'counters': self._metrics.get_counters(), 'stats': {
            'qualify_duration': self._metrics.get_stats('qualify_duration'),
            'batch_duration': self._metrics.get_stats('batch_duration'), },
        'cache': cache_stats,
        'circuit_breaker': {
            'state': self._circuit_breaker.state.value,
```

```
            'config': {
                'failure_threshold': self._circuit_breaker.config.failure_threshold,
                'recovery_timeout': self._circuit_breaker.config.recovery_timeout                }            },
'library_available': self._is_library_available()        }
    def get_library_info(self) -> Dict[str, Any]:
        """Получение информации о библиотеке."""
        if self._library_qualifier:
            try:
                info = self._library_qualifier.get_model_info() if hasattr(
                    self._library_qualifier, 'get_model_info'  ) else {}
                return {
                    'available': LIBRARY_AVAILABLE,
                    'loaded': self._library_qualifier.is_loaded,
                    'version': LIB_VERSION,
                    'device': self.config.device,
                    'model_path': self.config.model_path,
                    'info': info                }
            except Exception as e:
                logger.error(f"Error getting library info: {e}")
        return {
            'available': LIBRARY_AVAILABLE,
            'loaded': False,
            'device': self.config.device,
            'model_path': self.config.model_path,
            'version': LIB_VERSION        }
    def reset_circuit_breaker(self):
        """Сброс circuit breaker."""
        self._circuit_breaker.reset()
        logger.info("Circuit breaker reset")
    def clear_cache(self):
        """Очистка кэша."""
        self._cache.clear()
        logger.info("Cache cleared")
    def shutdown(self):
        """Graceful shutdown."""
        if self._shutdown_called:    return
        self._shutdown_called = True
        logger.info("Shutting down HQPAdapter...")
        if hasattr(self, '_async_pool'):
            self._async_pool.shutdown(wait=True)
        if hasattr(self, '_cache'):
            self._cache.clear()
        if hasattr(self, '_library_qualifier'):
            del self._library_qualifier
        try:
            import torch
            if torch.cuda.is_available():
                torch.cuda.empty_cache()
        except ImportError:
            pass
        logger.info("HQPAdapter shutdown complete")
```

ПРИЛОЖЕНИЕ Е

Сводные результаты тестирования гибридной модели

Для обеспечения целостного представления результатов экспериментального тестирования разработанной гибридной модели все полученные метрики сведены в единую таблицу. На ее основании могут быть сделаны выводы в разделе «5.2 Экспериментальная оценка качества квалификации контента».

Набор данных	**Метрика**	**Гибридная модель**	**DistilBERT**	**Разница**
Специализированный набор данных (высокий порог сарказма)	*Точность*	*94,22%*	*88,82%*	*+5,40 п.п.*
	F1-weighted	*95,01%*	*91,54%*	*+3,47 п.п.*
	Precision-weighted	96,22%	95,99%	+0,23 п.п.
	Полнота-weighted	94,22%	88,82%	+5,40 п.п.
	F1-macro	73,18%	65,22%	+7,96 п.п.
	Precision-macro	68,49%	61,40%	+7,09 п.п.
	Полнота-macro	82,46%	84,37%	-1,91 п.п.
Синтетический набор данных	*Точность*	*42,57%*	*42,00%*	*+0,57 п.п.*
	F1-weighted	*40,00%*	*40,00%*	*0,00 п.п.*
	Время обучения	1,4 сек	600 сек	в 440,9 раз
	Размер модели	8200 МБ	250 МБ	DistilBERT меньше
	Параметры	1,64 млн	66 млн	в 40,2 раз
	GPU	Нет	Да	Гибридная
Customer Support Ticket Dataset (Kaggle) Переведенный на русский язык	*Точность*	*22,58%*	*22,58%*	*0,00 п.п.*
	F1-score (weighted)	*15,1%*	*0*	*+ 15,1 п.п.*
	Интерпретируемость	Высокая	Низкая	Гибридная
	Время обучения	0,3 сек	18,0 сек	в 71,3 раз
	Размер модели	~15 МБ	255,4 МБ	в 17,0 раз
	Параметры	1,64 млн	66 млн	в 40,8 раз
	GPU	Нет	Да	Гибридная