МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ М. В. ЛОМОНОСОВА
ФАКУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ
КАФЕДРА МЕТОДОВ МАТЕМАТИЧЕСКОГО ПРОГНОЗИРОВАНИЯ

Голубев Роман Евгеньевич

Ускорение агентских приложений с помощью библиотеки Autointent

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
профессор, д.ф.-м.н.
Воронцов Константин Вячеславович

Москва, 2026

Оглавление

Введение 3
1 Постановка задачи 4
1.1 Граф диалога 4
1.2 Задача классификации интентов 4
1.3 Два подхода к классификации 4
1.4 Двухэтапная схема внедрения 5
2 Обзор предметной области 6
2.1 Агентские диалоговые системы 6
2.2 Классификация интентов 7
2.3 Автоматическое машинное обучение (AutoML) 8
2.4 Библиотека AutoIntent 9
2.5 Библиотека dialog2graph 10
2.6 PydanticAI 10
3 Описание системы и методов 11
3.1 Общая архитектура экспериментальной системы . . . 11
3.2 Формализация задачи классификации интентов 12
3.3 Классификаторы 13
3.4 Генерация данных и протокол эксперимента 15
3.5 Методика оценки 15
4 Вычислительные эксперименты 17
4.1 Краткое описание постановки 17
4.2 Результаты: режим edge (верхняя оценка) 17
4.3 Результаты: режим llm (парафразированные сообщения) 18
4.4 Анализ латентности 19
4.5 Основные выводы 19
5 Заключение 20

Список литературы **22**

Введение

Диалоговые системы и чат-боты стали неотъемлемой частью современного бизнеса: они обрабатывают обращения в службах поддержки, помогают пользователям ориентироваться в сложных сервисах, сокращают нагрузку на операторов колл-центров. С развитием больших языковых моделей (Large Language Models, LLM) возможности таких систем существенно расширились: появились так называемые агентские приложения, в которых модель не просто генерирует ответ, но и самостоятельно выбирает, какое действие или инструмент следует задействовать.

Ключевым компонентом агентской диалоговой системы является классификатор интентов — модуль, определяющий намерение пользователя и выбирающий подходящий следующий шаг сценария. Традиционно эту роль берёт на себя большая языковая модель, что удобно, поскольку не требует предварительного обучения на размеченных данных. Однако большая языковая модель — дорогостоящий и относительно медленный инструмент: каждый вызов занимает от одной до нескольких секунд и влечёт финансовые издержки, пропорциональные числу обрабатываемых запросов.

Альтернативный подход состоит в замене языковой модели на компактный специализированный классификатор, обученный непосредственно на диалоговых данных конкретного приложения. Такой классификатор может работать в разы быстрее, при этом сохраняя качество принятия решений, достаточное для практического применения. Однако, реализация подобного подхода требует, накопления обучающих примеров — пар «контекст диалога → выбранный переход» — что возможно только после развёртывания системы в рабочей среде.

В настоящей работе рассматривается способ ускорения агентских диалоговых приложений посредством замены LLM-классификатора интентов на лёгкие модели, получаемые с помощью инструментов автоматического машинного обучения, реализованных в библиотеке Autointent [1]. Целью данной работы является исследование и демонстрация эффективности предложенного метода ускорения по сравнению с классическим подходом.

1. Постановка задачи

1.1. Граф диалога

Пусть структура диалогового сценария задана в виде *графа диалога* $G = (V, E)$, где V — конечное множество узлов (состояний бота), $E \subseteq V \times V$ — множество направленных рёбер (возможных переходов), и выделен начальный узел $v_0 \in V$.

Каждый узел $v \in V$ снабжён меткой $\ell(v)$ — кратким описанием состояния диалога (например, «Greeting», «Clarify issue», «Suggest fix»), а также набором *эталонных реплик бота* U_v^{bot}, типичных для этого состояния. Каждое ребро $e = (v_s, v_t) \in E$ снабжено набором *эталонных реплик пользователя* U_e^{user}, иллюстрирующих тот интент, который провоцирует переход $v_s \to v_t$.

1.2. Задача классификации интентов

В ходе диалога система в каждый момент времени находится в некотором узле $v_s \in V$. Наблюдается *контекст диалога* — последовательность чередующихся реплик бота и пользователя, заканчивающаяся очередной репликой пользователя u. Требуется определить, по какому ребру $e^* = (v_s, v_t)$ следует совершить переход, то есть классифицировать реплику u в один из интентов, соответствующих исходящим рёбрам узла v_s.

Формально задача классификации интентов в состоянии v_s при реплике пользователя u и контексте c записывается как

$$e^* = \arg \max_{e=(v_s,v_t)\in E} f_\theta(u, c, e), \tag{1.1}$$

где f_θ — классификатор с параметрами θ, оценивающий релевантность ребра e данному контексту.

1.3. Два подхода к классификации

В настоящей работе сравниваются два подхода к реализации функции f_θ.

LLM-классификатор (базовый). Функцию f_θ реализует большая языковая модель. На вход подаётся промпт, содержащий: тему диалога, текущее

состояние бота, реплику бота, реплику пользователя, а также список кандидатных рёбер с их метками и примерами реплик. Модель возвращает идентификатор наиболее подходящего ребра. Данный подход не требует обучения, однако характеризуется высокой задержкой и стоимостью вызова.

Классификатор Autointent (предлагаемый). Функцию f_θ реализует компактная специализированная модель, обученная с помощью библиотеки Autointent [1]. Обучающей выборкой служат пары (u_i, e_i), где u_i — реплика пользователя, e_i — соответствующее ребро. При инференсе модель работает локально без обращения к внешнему API, что обеспечивает задержку на два порядка меньше, чем у LLM.

1.4. Двухэтапная схема внедрения

Предлагаемая схема внедрения Autointent в действующее агентское приложение состоит из двух стадий.

Стадия 1 (накопление данных). Агентское приложение работает в штатном режиме: классификацию выполняет LLM. Параллельно активируется *сборщик данных* — программный модуль, который перехватывает все решения классификатора и записывает в журнал пары «реплика пользователя → выбранное ребро». На этой стадии качество системы не снижается, а данные накапливаются органически в ходе штатной эксплуатации.

Стадия 2 (переключение на лёгкую модель). После накопления достаточного объёма данных с помощью Autointent обучается компактный классификатор. Если достигнутое качество не ниже порогового, LLM-классификатор заменяется новой моделью. Переключение происходит без изменения бизнес-логики приложения — достаточно поменять один параметр конфигурации.

2. Обзор предметной области

2.1. Агентские диалоговые системы

Агентские приложения представляют собой программные системы, в которых языковая модель используется не только для генерации текста, но и для управления ходом диалога, выбора следующего действия и обращения к внешним инструментам [2, 3]. В отличие от традиционных диалоговых систем, где сценарий взаимодействия обычно задаётся жёстко, агентский подход допускает динамическое принятие решений на основе текущего контекста, истории диалога и доступных инструментов.

Одним из наиболее распространённых способов повышения качества таких систем является архитектура Retrieval-Augmented Generation (RAG) [4]. В этой схеме генеративная модель дополняется внешней нереляционной памятью в виде поискового индекса или корпуса документов: сначала выполняется поиск релевантных фрагментов, а затем ответ формируется с учётом найденного контекста. Оригинальная работа по RAG показала, что такой подход позволяет явно обращаться к внешним знаниям, улучшать фактическую точность ответов и делать процесс более интерпретируемым за счёт использования опорных документов. Позднейшие обзорные работы по RAG показывают, что направление быстро развилось от базовой схемы retrieval+generation к более сложным вариантам, где отдельно исследуются качество поиска, стратегия агрегации контекста и способы контроля над использованием внешних знаний.

Для задач прикладного диалога RAG особенно важен тем, что он уменьшает зависимость модели от параметрических знаний, “запомненных” в весах. Это позволяет обновлять базу знаний без переобучения самой языковой модели и снижает риск ответа вне актуального контекста. Однако в агентской системе RAG обычно не является единственным вычислительным узлом: перед поиском или генерацией часто требуется определить, какой именно сценарий диалога следует активировать, к какому инструменту обращаться и какой тип ответа ожидается от пользователя.

Этап классификации интента является узким местом по задержке в тех случаях, когда он реализован с помощью LLM: поиск в индексе занимает единицы миллисекунд, тогда как вызов языковой модели занимает от одной

до нескольких секунд.

2.2. Классификация интентов

Задача классификации интентов в диалоговых системах изучается давно и остаётся одной из базовых задач task-oriented dialogue [5]. Классические методы, включая SVM, логистическую регрессию и другие линейные модели, обеспечивают высокую скорость инференса, но обычно требуют ручного проектирования признаков и хуже масштабируются на новые домены. Переход к нейросетевым моделям на основе трансформеров значительно улучшил качество классификации и снизил зависимость от ручного feature engineering [6].

С появлением instruction-tuned языковых моделей распространился zero-shot и few-shot подход к классификации интентов: модели передаются описания классов или несколько примеров, после чего она выбирает наиболее подходящий класс без дополнительного обучения. Такая схема удобна на этапе быстрого прототипирования и при отсутствии размеченных данных, однако её применение в рабочем контуре ограничивается вычислительной стоимостью, задержкой ответа и вариативностью поведения. Исследования zero- и few-shot intent classification показывают, что LLM-подход действительно может решать задачу в условиях нехватки данных, но при этом его преимущества наиболее заметны именно как инструмента для cold-start и генерации начальной разметки, а не как оптимального runtime-решения [**zeroshot_intent_llm**].

Таким образом, в практических диалоговых системах возникает компромисс между качеством и вычислительной эффективностью. LLM-классификация удобна как универсальный и гибкий механизм, особенно на ранних этапах разработки, однако для фиксированного набора интентов и заранее известного графа диалога более предпочтительным становится компактный классификатор, который можно быстро вызывать в онлайне. Именно в такой постановке задача сводится к поиску модели, которая сохраняет качество LLM-подхода, но обеспечивает существенно меньшую задержку и более предсказуемое поведение.

С появлением инструкционно-настроенных LLM возник подход zero-shot классификации [7]: модели передаются описания классов в промпте, и она напрямую выбирает подходящий. Этот подход не требует размеченных данных, однако страдает высокой задержкой и стоимостью.

В настоящей работе исследуется подход, при котором zero-shot LLM используется лишь на начальном этапе для накопления данных, а затем заменяется компактным классификатором, обученным с помощью AutoML.

2.3. Автоматическое машинное обучение (AutoML)

Автоматическое машинное обучение (AutoML) представляет собой класс методов, направленных на автоматизацию ключевых этапов построения моделей машинного обучения, включая выбор архитектуры модели, настройку гиперпараметров, разбиение данных и оценку качества [8].

Современные AutoML-системы позволяют существенно снизить требования к экспертной квалификации пользователя и обеспечивают возможность построения конкурентоспособных моделей в режиме “машинного обучения как сервиса” (ML-as-a-Service). Данный подход особенно востребован в прикладных задачах анализа текста, включая классификацию, анализ тональности и тематическое моделирование.

В зависимости от уровня автоматизации, AutoML можно разделить на несколько направлений.

Табличный AutoML ориентирован на задачи с табличными данными и включает автоматизацию feature engineering, отбора признаков и ансамблирования моделей. В таких системах часто используются классические алгоритмы машинного обучения, такие как градиентный бустинг и линейные модели, а оптимизация гиперпараметров осуществляется с помощью бюджет-ориентированных методов, например Bayesian optimization или Tree-structured Parzen Estimator.

Нейро-ориентированный AutoML включает методы нейросетевого поиска архитектур (Neural Architecture Search, NAS), которые автоматизируют выбор структуры глубокой модели. Несмотря на высокую эффективность, NAS обычно решает только задачу выбора архитектуры и не охватывает полный цикл AutoML, включая подготовку данных и оптимизацию пайплайна.

AutoML для обработки естественного языка отличается использованием трансформерных моделей и эмбеддинговых представлений текста. В отличие от классических табличных задач, здесь значительная часть пайплайна заключается в выборе предобученной языковой модели и способа её использования в качестве признакового пространства.

Существуют три основных стратегии автоматизации в NLP AutoML. Первая из них предполагает использование жёстко фиксированных пресетов гиперпараметров, которые обеспечивают разумный баланс между качеством модели и вычислительными затратами. Вторая стратегия основана на полном поиске гиперпараметров с применением методов оптимизации, таких как Bayesian optimization, что позволяет более точно подстраивать модель под конкретную задачу. Наконец, третья стратегия связана с мета-обучением, при котором система предсказывает наиболее подходящие конфигурации моделей на основе характеристик набора данных.

Несмотря на прогресс, существующие AutoML-решения для NLP часто ограничены поддержкой базовых задач классификации и не всегда учитывают специфические требования диалоговых систем, такие как обработка out-of-scope (OOS) запросов и мульти-лейбловая классификация.

2.4. Библиотека AutoIntent

AutoIntent [1] представляет собой систему автоматизированного машинного обучения, разработанную для задач классификации интентов в диалоговых системах. В отличие от общих AutoML-фреймворков, AutoIntent ориентирован на специфику обработки естественного языка и сценариев разговорного искусственного интеллекта.

Ключевой особенностью AutoIntent является end-to-end автоматизация пайплайна классификации текстов, включающая автоматический выбор модели эмбеддингов, алгоритма классификации и порогов принятия решений. Такой подход позволяет рассматривать весь процесс построения классификатора как единую задачу оптимизации и обеспечивает более согласованную настройку его компонентов.

Архитектурно AutoIntent реализует sklearn-подобный интерфейс и поддерживает широкий набор моделей, включая линейные классификаторы, метрические методы, fine-tuning трансформеров и zero-shot подходы.

Отдельным компонентом системы является модуль принятия решений, который преобразует вероятностные оценки в итоговые предсказания.

AutoIntent реализует иерархическую оптимизацию пайплайна, включающую последовательный подбор модели эмбеддингов, классификатора и параметров decision layer.

2.5. Библиотека dialog2graph

Библиотека dialog2graph [9] предоставляет инструменты для работы с диалоговыми графами: их загрузки, обхода, визуализации и генерации синтетических диалогов.

2.6. PydanticAI

PydanticAI [10] — фреймворк для построения агентских приложений на Python с типобезопасным описанием инструментов. В настоящей работе он используется для реализации ReAct-агента, в котором классификатор интентов подключается как обычный инструмент.

3. Описание системы и методов

В данной работе исследуется задача классификации интентов в фиксированном графе диалога. Каждое переходное ребро графа рассматривается как отдельный класс (интент), а задача классификации сводится к выбору корректного перехода $e = (v_s, v_t)$ на основе локального контекста диалога.

Основная цель экспериментов заключается в сравнении двух подходов к решению данной задачи: (1) классификация с использованием большой языковой модели (LLM), (2) классификация с использованием модели, автоматически подобранной с помощью AutoML (Autointent [1]).

Сравнение проводится по качеству предсказаний и вычислительной эффективности.

3.1. Общая архитектура экспериментальной системы

Экспериментальная система построена вокруг формализованного представления диалога в виде ориентированного графа $G = (V, E)$, где вершины V соответствуют состояниям диалоговой системы, а рёбра E задают допустимые переходы между состояниями и интерпретируются как пользовательские интенты. Каждому узлу и ребру сопоставлены текстовые реплики, используемые при генерации диалогов и обучении моделей.

Ключевым компонентом системы является симулятор диалога, который формирует синтетические диалоговые траектории путём случайного обхода графа. На каждом шаге выбирается одно из исходящих рёбер текущего узла, что определяет следующий переход, а соответствующие текстовые поля используются для построения реплик ассистента и пользователя. Пользовательские сообщения могут формироваться как из заранее заданных примеров, так и с помощью языковой модели, генерирующей парафразы, что позволяет повысить разнообразие входных данных.

Для решения задачи классификации интентов используются два альтернативных подхода. Первый основан на применении большой языковой модели, которая в zero-shot режиме ранжирует возможные переходы, используя текстовое описание текущего состояния диалога и список кандидатов. Второй подход реализуется с помощью специализированного классификатора, обучаемого средствами автоматического машинного обучения в библиоте-

ке Autointent. Обучение данного классификатора производится на примерах, извлечённых непосредственно из структуры графа, где каждому пользовательскому высказыванию сопоставляется идентификатор соответствующего ребра.

Оценка качества работы классификаторов выполняется на сгенерированных диалогах с известной целевой разметкой. Для каждого шага фиксируется корректный интент, определяемый выбранным переходом в графе, после чего вычисляются метрики качества и временные характеристики инференса для каждого из рассматриваемых подходов.

Дополнительно в системе присутствует модуль формирования выборок, который обеспечивает построение обучающих данных для AutoIntent и генерацию тестовых диалогов. При этом обучающая выборка формируется исключительно на основе структуры графа и не зависит от предсказаний языковой модели, что позволяет корректно сравнивать оба подхода в условиях единой целевой разметки.

3.2. Формализация задачи классификации интентов

Пусть задан ориентированный граф диалога $G = (V, E)$, в котором вершины V соответствуют состояниям диалоговой системы, а рёбра E задают допустимые переходы между ними. Каждое ребро $e = (v_s, v_t) \in E$ интерпретируется как отдельный интент пользователя и снабжается уникальным идентификатором $\mathrm{id}(e)$.

На каждом шаге диалога система располагает информацией о текущем состоянии v_s, последней реплике ассистента a_{t-1} и текущей реплике пользователя u_t. Эти величины образуют наблюдаемый контекст, на основании которого необходимо определить дальнейшее развитие диалога.

Задача классификации интентов формализуется как построение отображения

$$f : (v_s, a_{t-1}, u_t) \to e \in E, \tag{3.1}$$

которое сопоставляет наблюдаемому контексту одно из допустимых исходящих рёбер из вершины v_s. Таким образом, множество возможных классов на каждом шаге ограничено подмножеством рёбер, инцидентных текущей вершине, что отличает рассматриваемую постановку от классической задачи многоклассовой классификации с фиксированным набором классов.

3.3. Классификаторы

LLM-классификатор

LLM-классификатор реализует zero-shot подход к задаче выбора интента, при котором языковая модель используется непосредственно на этапе инференса без предварительного обучения на целевых данных. На вход модели подаётся текстовый промпт, содержащий описание текущего состояния диалога, последнюю реплику ассистента, текущую реплику пользователя, а также полный список допустимых переходов из текущего состояния.

Каждое допустимое ребро $e = (v_s, v_t)$ кодируется в текстовом виде с использованием его идентификатора, целевого состояния и набора примеров пользовательских реплик, ассоциированных с данным переходом. Формально представление кандидата имеет следующий вид:

```
id: <src_label>__to__<tgt_label>
target_state: <tgt_label>
example_user_utterances:
    - ...
```

Сформированный таким образом список кандидатов передаётся языковой модели вместе с контекстом диалога. Модель выполняет ранжирование возможных переходов и возвращает идентификатор наиболее вероятного интента либо список из K наиболее вероятных кандидатов. Таким образом, выбор следующего шага диалога осуществляется непосредственно за счёт обобщающей способности языковой модели, без использования специализированного обученного классификатора.

AutoML-классификатор (Autointent)

Второй подход основан на использовании библиотеки Autointent [1], реализующей методы автоматического машинного обучения для задач текстовой классификации. В отличие от LLM-подхода, данный классификатор обучается на специально сформированной выборке и используется в режиме быстрого инференса без обращения к языковой модели.

Обучающая выборка формируется непосредственно на основе структуры диалога, заданного графом. Для каждого ребра $e = (v_s, v_t)$ и каждой

пользовательской реплики $u \in U_e$ создаётся обучающий пример, представленный в виде пары

$$(u, \mathrm{id}(e)). \tag{3.2}$$

Таким образом, каждое ребро графа интерпретируется как отдельный класс, а задача классификации сводится к многоклассовой классификации текстовых входов, где необходимо сопоставить пользовательскую реплику одному из возможных переходов в графе.

Обучение. В процессе обучения Autointent автоматически осуществляет подбор компонентов пайплайна классификации. В частности, выбирается модель для построения эмбеддингов текстовых представлений, алгоритм классификации в пространстве эмбеддингов, а также параметры, определяющие пороги принятия решений. Оптимизация данных компонентов выполняется в рамках процедур автоматического машинного обучения с использованием методов Bayesian optimization, обеспечивающих поиск конфигурации, оптимизирующей целевую метрику качества классификации.

Инференс. На этапе инференса входной контекст диалога преобразуется в последовательность структурированных сообщений, включающих информацию о текущем состоянии графа, теме диалога, последней реплике ассистента и текущей реплике пользователя. Такое представление формализуется следующим образом:

```
[assistant] graph_state: <state>
[assistant] topic: <topic>
[assistant] <last assistant message>
[user] <user message>
```

Полученный контекст передаётся в модель Autointent, которая возвращает ранжированный список наиболее вероятных интентов, соответствующих возможным переходам в графе диалога.

На основе этого контекста модель возвращает ранжированный список интентов.

3.4. Генерация данных и протокол эксперимента

Для оценки качества классификаторов используется синтетический датасет, формируемый на основе исходных графов диалога. Такой подход позволяет контролировать структуру диалогов и обеспечивать однозначное соответствие между наблюдаемым контекстом и целевым интентом, заданным графом.

Генерация диалогов. Диалоги формируются процедурой случайного обхода графа. В начальный момент выбирается стартовое состояние, после чего на каждом шаге осуществляется переход по одному из исходящих рёбер текущей вершины. Каждое такое ребро определяет целевой интент и, соответственно, правильный ответ классификатора.

Пользовательская реплика для каждого перехода генерируется языковой моделью и рассматривается как семантическая парафраза эталонных utterances, ассоциированных с соответствующим ребром. Таким образом, обеспечивается вариативность формулировок при сохранении неизменной семантики интента.

Формирование выборок. Сгенерированные последовательности переходов преобразуются в набор независимых обучающих примеров вида «контекст диалога – целевой интент» и далее разделяются на обучающую и тестовую выборки. Разбиение выполняется таким образом, чтобы обеспечить отсутствие пересечения между примерами в train и test частях.

Модель AutoML (Autointent) обучается исключительно на обучающей части данных, тогда как оценка качества обеих систем — LLM-классификатора и обученного AutoML-классификатора — проводится только на тестовой выборке. Такой протокол исключает утечку информации и обеспечивает корректное сравнение подходов в одинаковых условиях.

3.5. Методика оценки

Сравнение методов проводится по двум группам метрик:

Метрики качества классификации:

- Accuracy@1 — точность топ-1 предсказания;
- Accuracy@K — попадание правильного интента в топ-K;
- MRR@K — mean reciprocal rank;
- macro-averaged Accuracy@1 — усреднение по классам;
- invalid prediction rate — доля предсказаний вне множества допустимых рёбер.

Метрики диалогового уровня:

- dialog all-correct@K — доля диалогов, в которых все шаги предсказаны корректно;

Метрики производительности:

- средняя и медианная задержка инференса;
- распределение времени ответа классификаторов.

Такой протокол позволяет одновременно оценивать качество, устойчивость на уровне диалога и вычислительную эффективность методов.

4. Вычислительные эксперименты

4.1. Краткое описание постановки

Эксперименты проводились на подмножестве датасета `voorhs/d2g_-generated`, содержащего диалоговые графы. Каждый граф интерпретируется как конечный автомат диалога, где узлы соответствуют состояниям, а рёбра — пользовательским интентам.

Для оценки были выбраны три графа с наибольшим числом рёбер при условии $n_{\text{edges}} \geqslant 30$: $\{245, 62, 238\}$.

Рассматривались два режима генерации пользовательских сообщений:

- **edge** — упрощённый режим (верхняя оценка), в котором пользовательские сообщения совпадают с примерами из графа;
- **llm** — более реалистичный режим, в котором пользовательские сообщения генерируются LLM как парафразы.

Сравнивались два подхода:

- **AutoIntent** — обучаемый классификатор интентов;
- **LLM-baseline** — ранжирование кандидатов с помощью LLM без обучения.

Оценивание выполнялось на синтетических многоходовых диалогах, полученных случайным обходом графа.

4.2. Результаты: режим edge (верхняя оценка)

Таблица 1. Результаты в режиме edge (пользовательские сообщения из графа).

Граф	Ходов	AI Acc@1	LLM Acc@1	AI Dialog@1	LLM Dialog@1	LLM p50 (мс)
62	94	1.0000	1.0000	1.0000	1.0000	1178.8
238	114	1.0000	0.9825	1.0000	0.9333	1109.8
245	132	1.0000	1.0000	1.0000	1.0000	1032.6
Среднее	–	1.0000	0.9942	1.0000	0.9778	1107.1

Анализ результатов. В данном режиме задача существенно упрощается, поскольку пользовательские сообщения фактически совпадают с обучающими примерами.

AutoIntent демонстрирует идеальное качество (Accuracy@1 = 1.0) на всех графах. LLM-baseline также показывает близкие результаты, однако наблюдаются отдельные ошибки (например, на графе 238).

При этом различие по времени отклика является существенным: AutoIntent работает за миллисекунды, тогда как LLM требует порядка одной секунды на один ход.

4.3. Результаты: режим llm (парафразированные сообщения)

Таблица 2. Результаты в режиме llm (парафразированные пользовательские сообщения).

Граф	Ходов	AI A@1	LLM A@1	AI D@1	LLM D@1	LLM p50 (мс)
62	47	0.8085	0.8298	0.4000	0.4667	1065.1
238	59	1.0000	0.9661	1.0000	0.8667	999.2
245	75	0.9200	0.9467	0.8000	0.8000	949.7
Среднее	–	0.9095	0.9142	0.7333	0.7111	1004.7

Анализ результатов. В более реалистичном режиме (с парафразированием) качество классификации снижается для обоих методов.

AutoIntent и LLM-baseline показывают сопоставимые результаты:

- средняя Accuracy@1: 0.91 (AutoIntent) против 0.91 (LLM),
- различия между графами существенны (например, граф 62 значительно сложнее).

При этом заметно падение метрики **dialog all-correct@1**, что указывает на накопление ошибок в многоходовых диалогах.

4.4. Анализ латентности

- AutoIntent: медианная задержка (p50) составляет примерно 2–4 мс;
- LLM-baseline: медианная задержка (p50) составляет примерно 950–1200 мс;
- p95 для LLM достигает 1.5–1.9 секунды.

Вывод по производительности. AutoIntent обеспечивает ускорение примерно на два порядка (100–500x) по сравнению с LLM-baseline, что делает его существенно более пригодным для использования в реальных интерактивных системах.

4.5. Основные выводы

- В упрощённом режиме (edge) AutoIntent достигает идеального качества.
- В реалистичном режиме (llm) AutoIntent и LLM демонстрируют сопоставимую точность.
- Многоходовая оценка выявляет существенное накопление ошибок на уровне диалога.
- AutoIntent значительно превосходит LLM по скорости (на 2–3 порядка).

Обсуждение. Полученные результаты показывают, что обучаемые модели интентов могут эффективно конкурировать с LLM-подходами по качеству, при этом обеспечивая значительно более низкую латентность. Это делает их перспективными для применения в системах, требующих быстрых и стабильных ответов (например, чат-ботах и голосовых ассистентах).

5. Заключение

В рамках выпускной квалификационной работы была исследована задача классификации пользовательских интентов в агентских диалоговых системах, основанных на графах диалога, и предложен эффективный подход к её решению с использованием компактного обучаемого классификатора.

Основные результаты работы заключаются в следующем.

1) Сформулирована задача классификации интентов в графах диалога с учётом того, что множество допустимых интентов динамически определяется текущим состоянием. Введена формализация, в которой интенты соответствуют рёбрам графа, а задача классификации сводится к выбору корректного перехода на каждом шаге диалога из множества допустимых.

2) Предложен подход к интеграции компактного классификатора интентов (AutoIntent) в агентскую систему, позволяющий заменить LLM-классификатор без изменения бизнес-логики. Рассмотрена двухэтапная схема внедрения, включающая сбор обучающих данных с использованием LLM и последующее обучение специализированной модели.

3) Реализована система экспериментального оценивания, включающая генерацию синтетических многоходовых диалогов на основе случайного обхода графа, поддержку различных режимов пользовательских сообщений (в том числе с парафразированием при помощи LLM), а также единый интерфейс для сравнения AutoIntent и LLM-baseline. В систему оценки включён расширенный набор метрик, учитывающий как точность классификации, так и поведение модели на уровне диалога и характеристики латентности.

4) Проведены вычислительные эксперименты на нескольких графах диалога представляющих наиболее сложные случаи по числу рёбер. Показано, что в упрощённом режиме (edge) AutoIntent достигает предельного качества (Accuracy@1 = 1.0), тогда как в более реалистичном режиме (llm) его качество остаётся сопоставимым с LLM-baseline (в среднем около 0.91). При этом установлено, что в многоходовых диалогах проис-

ходит накопление ошибок, что приводит к снижению метрик на уровне диалога.

5) Установлено, что использование AutoIntent обеспечивает существенное снижение времени отклика: медианная задержка составляет единицы миллисекунд против порядка одной секунды у LLM, что соответствует ускорению на 2–3 порядка. Таким образом, предложенный подход позволяет сохранить сопоставимое качество классификации при радикальном выигрыше в производительности.

Таким образом, в работе показано, что специализированные классификаторы интентов, учитывающие структуру диалогового графа, могут эффективно заменить LLM-классификаторы на этапе инференса без потери качества и с существенным снижением вычислительных затрат.

Перспективы дальнейших исследований. Перспективными направлениями дальнейшей работы являются расширение экспериментов на большее количество графов и предметных областей, исследование устойчивости модели к шумным пользовательским сообщениям, анализ зависимости качества от объёма обучающих данных, а также разработка гибридных стратегий, сочетающих использование AutoIntent и LLM в зависимости от сложности ситуации.

Список литературы

1. *Contributors A.* Autointent: AutoML for Intent Classification in Dialogue Systems. — 2024. — Дата обращения: 2025. `https://github.com/AutoIntentDev/autointent`.
2. ReAct: Synergizing Reasoning and Acting in Language Models / S. Yao [и др.] // International Conference on Learning Representations (ICLR). — 2023.
3. Toolformer: Language Models Can Teach Themselves to Use Tools / T. Schick [и др.]. — 2023.
4. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks / P. Lewis [и др.] // Advances in Neural Information Processing Systems (NeurIPS). — 2020.
5. A Survey of Joint Intent Detection and Slot Filling Models in Natural Language Understanding / H. Weld [и др.] // ACM Computing Surveys. — 2022. — Т. 55, № 8. — С. 1—38.
6. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin [и др.] // Proceedings of NAACL-HLT. — 2019. — С. 4171—4186.
7. Zero-shot User Intent Detection via Capsule Neural Networks / C. Xia [и др.] // Proceedings of EMNLP. — 2018. — С. 3090—3099.
8. *He X.*, *Zhao K.*, *Chu X.* AutoML: A Survey of the State-of-the-Art // Knowledge-Based Systems. — 2021. — Т. 212. — С. 106622.
9. *Contributors dialog2graph*. dialog2graph: Tools for Dialogue Graph Processing. — 2024. — Дата обращения: 2025. `https://github.com/voorhs/dialog2graph`.
10. *Pydantic*. PydanticAI: Agent Framework for Typed AI Applications. — 2024. — Дата обращения: 2025. `https://ai.pydantic.dev`.