САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Направление: 02.03.02
«Фундаментальная информатика и информационные технологии»

Основная образовательная программа: СВ.5190.2022
«Большие данные и распределенная цифровая платформа»

ОТЧЕТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Тема работы
*«*Система прогнозирования событий с использованием RAG*»*

Выполнил: Олизько Степан Сергеевич
Учебная группа
22.Б15-пу

Научный руководитель Антон Юрьевич Першин
Должность: старший преподаватель
Ученая степень: Ph. D.

Председатель комиссии: Алексей Юрьевич Утешев
Должность: профессор
Ученая степень: Доктор физико-математических наук

Санкт-Петербург
2024

Система прогнозирования событий с использованием RAG

Содержание

Введение

Системы прогнозирования событий на основе Retrieval-Augmented Generation (RAG) представляют собой современный подход к автоматизации аналитики и выработке обоснованных прогнозов. В основе RAG-архитектуры лежит комбинирование двух компонентов: модуля поиска и извлечения релевантной информации из внешних источников (корпусов текстов, баз данных, веб-ресурсов) и мощной языковой модели, способной на её основе генерировать связные выводы и прогнозы. Такой подход позволяет оперативно обновлять знания системы, опираясь не только на заранее обученные параметры, но и на свежие, контекстно важные данные.

В данной работе предлагается разработка системы прогнозирования событий, которая принимает на вход запросы пользователя на естественном языке, автоматически формирует поисковые запросы к внешним источникам, извлекает и фильтрует релевантные документы, а затем с помощью RAG-модуля строит и формулирует прогнозы. Архитектура системы включает компоненты для векторного поиска, ранжирования текстов, а также генеративную нейросеть, обогащённую контекстом из найденных фрагментов. Благодаря модульной структуре и механизму динамического обновления сущностной базы знаний система способна адаптироваться к новым тематикам и эффективно прогнозировать развитие событий в самых разных областях.

Цели и задачи исследования

Цель работы

Разработать и прототипировать систему прогнозирования событий на основе Retrieval-Augmented Generation, способную обрабатывать текстовые запросы, автоматически извлекать актуальную информацию из различных внешних источников и генерировать обоснованные прогнозы.

Задачи работы

1. Проанализировать существующие подходы к построению RAG-систем для прогнозирования событий.
2. Реализовать механизм поиска релевантной информации во внешних источниках.
3. Спроектировать архитектуру системы, включающую модули обработки, поиска и ранжирования документов для формирования релевантного контекста.
4. Интегрировать языковую модель в качестве генератора выводов, обеспечив передачу ей извлеченного контекста и управление параметрами прогноза.
5. Выполнить тестирование и оценку эффективности, сравнив качество прогнозов RAG-системы с прогнозами LLM и прогнозами человека.

Обзор научной литературы

1. Основы технологии Retrieval-Augmented Generation

Фундаментальные принципы RAG

Концепция Retrieval-Augmented Generation была впервые комплексно представлена в работе Lewis et al. (2020) "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" [1]. Авторы предложили гибридный подход к решению задач обработки естественного языка, требующих интенсивного использования знаний. Ключевая идея RAG заключается в объединении параметрической памяти предобученных языковых моделей с непараметрической памятью в виде внешних баз знаний, доступ к которым осуществляется через дифференцируемый механизм поиска.

Архитектура RAG состоит из двух основных компонентов:

- **Ретривер** $p_\eta(z|x)$ — возвращает наиболее релевантные документы для входного запроса x
- **Генератор** $p_\theta(y_i|x,z,y_{1:i-1})$ — создает выходную последовательность на основе входного запроса и извлеченных документов

В качестве ретривера используется Dense Passage Retriever (DPR), основанный на би-энкодерной архитектуре BERT, а генератор реализован на базе предобученной модели BART.

Lewis et al. выделяют два основных варианта реализации RAG-архитектуры:

- **RAG-Sequence** использует один и тот же набор извлеченных документов для генерации всей выходной последовательности, что обеспечивает согласованность контекста
- **RAG-Token** позволяет использовать различные документы для генерации каждого токена, что увеличивает гибкость модели и позволяет комбинировать информацию из нескольких источников

Экспериментальные результаты демонстрируют эффективность RAG-подхода на широком спектре задач: открытые вопросно-ответные системы (Natural Questions, TriviaQA, WebQuestions), абстрактивное реферирование (MS-MARCO), генерация вопросов в стиле Jeopardy и верификация фактов (FEVER). Особенно важным для задач прогнозирования является способность RAG генерировать более фактологически точные и специфичные ответы по сравнению с чисто параметрическими моделями.

Ключевое преимущество RAG-архитектуры заключается в возможности динамического обновления знаний системы путем замены индекса документов

без необходимости переобучения модели. Это особенно критично для систем прогнозирования событий, где актуальность информации играет решающую роль.

2. Современное состояние и развитие RAG-технологий

Систематический обзор современного состояния технологии RAG представлен в работе Gao et al. (2023) "Retrieval-Augmented Generation for Large Language Models: A Survey" [2]. Авторы определяют RAG как подход, объединяющий параметрические знания больших языковых моделей с обширными динамическими репозиториями внешних баз данных, что критически важно для систем прогнозирования событий, требующих актуальной информации.

В работе выделяются три основных парадигмы развития RAG:

Naive RAG — базовая архитектура, следующая традиционному процессу "индексирование-поиск-генерация". Для систем прогнозирования событий этот подход обеспечивает основу для извлечения релевантных исторических данных и их интеграции в процесс генерации прогнозов.

Advanced RAG — усовершенствованная версия с оптимизацией процессов до-поиска и пост-поиска. Включает стратегии улучшения качества индексирования через методы скользящего окна, детальной сегментации и добавления метаданных. Особую важность для прогнозирования представляют техники оптимизации запросов, включая переформулировку запросов, расширение запросов и их трансформацию.

Modular RAG — наиболее гибкая архитектура, поддерживающая как последовательную обработку, так и интегрированное обучение end-to-end. Включает специализированные модули:

- Search (адаптация к различным сценариям поиска)
- Memory (использование памяти LLM для направления поиска)
- Routing (навигация через различные источники данных)
- Predict (генерация контекста непосредственно через LLM)

Авторы детально анализируют три ключевых компонента RAG-систем, критичных для прогнозирования:

Компонент поиска (Retrieval) включает оптимизацию источников данных, granularity поиска и embedding-моделей. Для систем прогнозирования критически важны гибридные подходы, комбинирующие разреженный (BM25) и плотный (BERT-архитектура) поиск, а также fine-tuning embedding-моделей под специфические домены.

Компонент генерации (Generation) охватывает методы курирования контекста, включая re-ranking и сжатие контекста для решения проблемы "потери в середине" (lost in the middle). Особое внимание уделяется fine-tuning LLM для улучшения качества генерируемых прогнозов.

Процессы аугментации включают итеративный, рекурсивный и адаптивный поиск. Для прогнозирования событий особенно релевантен адаптивный подход (на примере FLARE и Self-RAG), где система автономно определяет необходимость дополнительного поиска информации.

Исследование выявляет ключевые проблемы RAG-систем, критичные для прогнозирования: проблемы точности поиска, галлюцинации в генерации и устойчивость к шуму. Авторы представляют комплексную систему оценки RAG-систем по критериям релевантности контекста, точности ответов, устойчивости к шуму и интеграции информации.

3. Применение RAG в задачах временного прогнозирования

RAG-архитектуры для прогнозирования временных событий

Значительный вклад в развитие методов прогнозирования временных событий с использованием больших языковых моделей внесла работа Chang et al. (2024) "A Comprehensive Evaluation of Large Language Models on Temporal Event Forecasting" [3]. Авторы провели комплексное исследование возможностей LLM в задачах прогнозирования событий, уделив особое внимание интеграции RAG-подходов.

Исследователи выделили три основных формата представления временных событий для LLM:

- **Графовый (structured)** — каждое атомарное событие представляется в виде четверки (субъект, отношение, объект, время), что соответствует формализации временных графов знаний (Temporal Knowledge Graphs, TKG)
- **Текстовый (unstructured)** — использует естественно-языковые описания событий, сохраняя детализированную контекстную информацию
- **Гибридный (graph-text hybrid)** — объединяет преимущества обоих подходов

Ключевым вкладом работы стала разработка и тестирование RAG-модулей для временного прогнозирования. Авторы предложили два основных метода построения исторического контекста:

Rule-based History — формирование контекста на основе предопределенных правил, учитывающих как глобальную перспективу (события с тем же субъектом), так и локальную (события в рамках одного комплексного события).

Retrieved History — динамический поиск релевантных исторических событий с использованием различных retrieval-моделей (BM25, Contriever, LlamaIndex).

Экспериментальные результаты показали, что RAG-подходы значительно улучшают качество прогнозирования в zero-shot режиме, однако не превосходят традиционные методы без дополнительной настройки. При этом fine-tuning LLM с интеграцией RAG демонстрирует наилучшие результаты. Особый интерес представляет обнаруженная авторами проблема popularity bias — системы показывают худшие результаты для редко встречающихся сущностей, что критично для практических приложений прогнозирования.

Авторы также создали специализированный dataset MidEast-TE-mini, включающий как структурированные события, так и исходные текстовые документы, что позволило провести сравнительный анализ различных подходов к представлению данных в RAG-системах.

Мульти-агентные RAG-системы с интеграцией графов знаний

Значительный вклад в развитие RAG-архитектур внесла работа Yu H.Q. и McQuade F. "RAG-KG-IL: A Multi-Agent Hybrid Framework for Reducing Hallucinations and Enhancing LLM Reasoning through RAG and Incremental Knowledge Graph Learning Integration" [4]. Авторы предложили фреймворк RAG-KG-IL (RAG-Knowledge Graph-Incremental Learning) — мульти-агентную гибридную систему для снижения галлюцинаций и повышения точности рассуждений больших языковых моделей.

Авторы выделяют ключевые ограничения существующих LLM-систем, критичные для задач прогнозирования:

- Неспособность эффективно работать со структурированными данными
- Статичность базы знаний
- Склонность к генерации фактически некорректной информации (галлюцинации)

Предложенный фреймворк RAG-KG-IL решает эти проблемы через интеграцию трех компонентов:

- **RAG-модуль** для извлечения верифицируемой информации из внешних источников
- **Граф знаний (Knowledge Graph)** для структурированного представления предметной области
- **Инкрементальное обучение** для динамического обновления базы знаний без полного переобучения

Экспериментальная оценка на медицинских запросах показала снижение количества галлюцинаций на 73% по сравнению с GPT-4o и значительное

улучшение полноты ответов. Особенно важным для систем прогнозирования является способность фреймворка к непрерывному расширению графа знаний: после обработки 20 вопросов система увеличила количество терминов с 57 до 226, число связей — с 114 до 420, а типов отношений — с 19 до 36.

Данный подход демонстрирует перспективность комбинирования RAG с структурированными представлениями знаний для создания адаптивных систем прогнозирования, способных к обучению в реальном времени и генерации более точных, обоснованных прогнозов.

4. Методология оценки и бенчмарки для систем прогнозирования

Формализация задачи прогнозирования как RAG-задачи

Прогнозирование будущих событий с использованием больших языковых моделей представляет собой активно развивающуюся область исследований. Традиционные подходы к созданию бенчмарков для оценки таких систем, включая работы Halawi et al. (2024), OpenEP (Guan et al., 2024) и ForecastBench (Karger et al., 2024), формализуют задачу прогнозирования как retrieval-augmented generation (RAG) задачу, где система должна сначала найти релевантные новостные статьи, а затем на их основе сформулировать прогноз.

Формально задача прогнозирования определяется как:

$$Y = \text{Reason}(Q, B, R, \text{Retrieve}(Q, X))$$

где:

- Q — вопрос для прогнозирования
- B — фоновая информация
- R — критерии разрешения
- X — корпус документов для поиска
- $Y \in [0,1]$ — предсказанная вероятность события

Для оценки качества прогнозов используется метрика Brier Score:

$$\text{Brier Score} = (1/N) \sum(Yn - \hat{Y}n)^2$$

Проблема инферируемости и метрика CIL

Ключевой проблемой существующих бенчмарков является отсутствие валидации инферируемости вопросов — то есть возможности получить обоснованный ответ на основе доступной информации. В работе Tao et al.

(2025) "PROPHET: An Inferable Future Forecasting Benchmark with Causal Intervened Likelihood Estimation" [5] впервые предложен систематический подход к решению этой проблемы через введение метрики Causal Intervened Likelihood (CIL).

Пусть $Y \in \{0,1\}$ обозначает, произойдет ли событие, описанное в вопросе, а $Xi \in \{0,1\}$ — произошло ли событие, описанное в i-й новостной статье. Каждая переменная Xi ассоциирована с датой Ti.

Определение CIL: $CILi = P(Y = \tilde{Y} \mid do(Xi = 1)) - P(Y = \tilde{Y} \mid do(Xi = 0))$

где do() — операция интервенции в каузальном выводе, а $\tilde{Y}$ — истинный ответ.

Для вычисления CIL авторы вводят три ключевых структурных предположения:

1. **Темпоральность:** $\forall i,j: Ti < Tj \Longrightarrow P(Xi \mid Xj) = P(Xi)$
2. **w-окно зависимости:** $\forall i,j: G(Xj) - G(Xi) > w \Longrightarrow (Xi, Xj) \notin$ edges of SCM
3. **Одновременная независимость:** $\forall i,j: G(Xj) = G(Xi) \Longrightarrow (Xi, Xj) \notin$ edges of SCM

где G(Xi) — индекс временной группы переменной Xi.

На основе этих предположений доказывается, что интервенционная вероятность может быть приближённо выражена через наблюдаемую:

$$P(Y = \tilde{Y} \mid do(Xi = 1)) \approx \sum P(Y = \tilde{Y} \mid Xi = 1, Xn_1, ...)\, P(Xn_1, ...)$$

где суммирование ведется по всем переменным $Xn_{\square}$, таким что $0 < G(Xi) - G(Xn_{\square}) \leq w$.

Предложенная методология CIL впервые обеспечивает количественную оценку инферируемости вопросов для прогнозирования, что критически важно для создания надежных бенчмарков. Экспериментальная валидация показала, что системы RAG демонстрируют значительно лучшую производительность на вопросах с высоким CIL по сравнению с вопросами с низким CIL, что подтверждает эффективность предложенного подхода.

Архитектура решения

1. Общая концепция системы

Разработанная система прогнозирования событий основана на парадигме Retrieval-Augmented Generation (RAG), которая объединяет возможности поиска релевантной информации с генеративными способностями языковых моделей. Основная идея заключается в том, что для получения качественных прогнозов система сначала извлекает актуальную информацию из внешних источников или предварительно подготовленных датасетов, а затем использует эту информацию для формирования обоснованного ответа.

Система работает по следующему принципу: пользователь задает вопрос о вероятности некоторого события, система автоматически определяет тематику запроса, формирует базу знаний одним из доступных способов, индексирует данные, находит наиболее релевантные фрагменты и передает их языковой модели для генерации итогового прогноза с указанием вероятности и обоснованием.

2. Компоненты системы

Формирование базы знаний

Система поддерживает два подхода к формированию базы знаний

Динамическое извлечение через API The Guardian:

Система автоматически извлекает актуальную информацию из новостных источников на основе тематики пользовательского запроса. Процесс включает:

- Определение ключевых слов из пользовательского запроса
- Выполнение поискового запроса через The Guardian API
- Извлечение полного содержимого статей по полученным ссылкам
- Фильтрация и очистка текстового контента

Использование предварительно подготовленного датасета PROPHET
Альтернативный подход использует специализированный датасет релевантных новостей PROPHET [5], который содержит структурированную коллекцию новостных материалов, предварительно размеченных и организованных для задач прогнозирования. Этот подход обеспечивает:

- Высокое качество и релевантность данных
- Предварительную обработку и структурирование информации
- Стабильность и воспроизводимость результатов
- Отсутствие зависимости от внешних API

Независимо от источника, полученные тексты обрабатываются единообразно - разбиваются на небольшие перекрывающиеся фрагменты (чанки).

Векторизация и индексирование

Текстовые фрагменты преобразуются в числовые векторные представления с помощью специальных моделей энкодеров. Система поддерживает два типа векторизации:

- **Плотная векторизация**: создание семантически богатых векторных представлений, которые отражают смысловое содержание текста
- **Разреженная векторизация**: статистическое представление на основе частоты терминов, эффективное для поиска по ключевым словам

Полученные векторы организуются в специальные индексы, обеспечивающие быстрый поиск наиболее похожих фрагментов.

Поиск и ранжирование

При поступлении запроса система преобразует его в векторное представление и ищет наиболее похожие фрагменты в индексах. Найденные кандидаты дополнительно переранжируются с помощью специальных моделей, которые более точно оценивают релевантность с учетом контекста.

Генерация ответа

Отобранные релевантные фрагменты объединяются с исходным запросом и передаются языковой модели, которая генерирует финальный ответ с указанием вероятности события и обоснованием прогноза.

3. Архитектурные варианты RAG-пайплайнов

Plain LLM (Базовая языковая модель)

Простейший подход, где языковая модель отвечает на вопрос, опираясь только на свои внутренние знания без дополнительного контекста. Служит базелайном для сравнения эффективности RAG-подходов.

Процесс работы:

1. Пользовательский запрос поступает напрямую в языковую модель
2. Модель генерирует ответ на основе предварительно обученных знаний
3. Возвращается прогноз с вероятностью события

Наивный RAG

Классическая реализация RAG с использованием одного типа векторного поиска. Обеспечивает базовые возможности извлечения релевантной информации.

Процесс работы:

1. Запрос преобразуется в векторное представление
2. Система формирует базу знаний одним из доступных способов:
    - Через API The Guardian с автоматическим определением тематики
    - Из датасета PROPHET с фильтрацией по релевантности
3. Данные векторизуются и индексируются
4. Выполняется поиск наиболее похожих фрагментов
5. Найденные фрагменты передаются языковой модели вместе с запросом
6. Генерируется итоговый прогноз

Гибридный RAG с Cross-Encoder переранжированием

Продвинутый подход, объединяющий преимущества разных типов поиска и использующий дополнительное переранжирование для повышения качества.

Процесс работы:

1. Запрос обрабатывается параллельно двумя типами энкодеров (плотный и разреженный)
2. Система формирует базу знаний выбранным способом (API или датасет)
3. Данные векторизуются обоими способами и индексируются
4. Выполняется гибридный поиск по обоим индексам
5. Результаты объединяются и переранжируются с помощью Cross-Encoder модели
6. Лучшие фрагменты передаются языковой модели для генерации ответа

Реализация

1. **Технологический стек**

Основные библиотеки и инструменты

Фреймворки для работы с языковыми моделями:

- **LangChain**: Основной фреймворк для создания RAG-пайплайнов, предоставляющий инструменты для разбиения документов, работы с векторными базами и интеграции с LLM
- **Groq**: Библиотека для работы с инференс провайдером groq.

Векторизация и поиск:

- **Sentence-Transformers**: Специализированная библиотека для создания семантических векторных представлений текста
- **FAISS (Facebook AI Similarity Search)**: Высокопроизводительная библиотека для поиска по векторам, оптимизированная для работы с большими массивами данных
- **scikit-learn**: Для реализации TF-IDF векторизации и других статистических методов

Работа с данными:

- **pandas**: Обработка структурированных данных и работа с датасетом PROPHET
- **numpy**: Численные вычисления и работа с векторными представлениями

Веб-запросы и парсинг:

- **aiohttp**: Асинхронная HTTP-библиотека для эффективной работы с внешними API
- **BeautifulSoup4**: Парсинг HTML-контента полученных веб-страниц
- **requests**: Синхронные HTTP-запросы

Утилиты:

- **Pydantic**: Валидация данных и управление конфигурацией
- **asyncio**: Асинхронное программирование для параллельной обработки

Модели и API

Языковые модели:

- **Llama-3.1-8b-instant** (через Groq API): Основная генеративная модель для создания прогнозов

Модели векторизации:

- **all-MiniLM-L6-v2**: Sentence-BERT модель для создания плотных векторных представлений
- **ms-marco-MiniLM-L-6-v2**: Cross-encoder модель для переранжирования

Источники данных:

- **The Guardian API**: Источник актуальных новостей для динамического построения контекста
- **PROPHET Dataset**: Предварительно подготовленный датасет релевантных новостей для прогнозирования

2. Детали реализации

Обработка и разбиение текстов

Функция разбиения текстов на чанки:

```
def chunk_texts(texts, metadatas=None, chunk_size=500, chunk_overlap=50):
        splitter = RecursiveCharacterTextSplitter(chunk_size=chunk_size,
chunk_overlap=chunk_overlap)
        chunks = splitter.create_documents(texts = texts, metadatas=metadatas)

        return chunks
```

Система векторизации

Плотная векторизация с использованием SBERT:

```
class SBERTEncoder:
        def __init__(self):
        self.model = SentenceTransformer("all-MiniLM-L6-v2")

        def encode(self, texts):
        return self.model.encode(texts, convert_to_numpy=True)
```

Модель all-MiniLM-L6-v2 была выбрана как компромисс между качеством векторных представлений и скоростью работы. Она создает 384-мерные векторы, эффективно кодирующие семантическое содержание текста.

Разреженная векторизация с TF-IDF:

```
class TfidfEncoder:
        def __init__(self):
        self.vectorizer = TfidfVectorizer()

        def fit(self, texts):
        self.vectorizer.fit(texts)

        def transform(self, texts):
        return self.vectorizer.transform(texts).toarray()

        def fit_transform(self, texts):
        return self.vectorizer.fit_transform(texts).toarray()
```

Индексирование с FAISS

```
class Indexer:
        def __init__(self, index_type="faiss"):
        self.index_type = index_type
        self.index = None
        self.vectors = None
        self.chunks = None  # сюда будем сохранять тексты

        def build(self, vectors, chunks):
        logger.info("Building index...")
        self.vectors = vectors
        self.chunks = chunks
        dim = vectors.shape[1] if hasattr(vectors, 'shape') else len(vectors[0])
        self.index = faiss.IndexFlatL2(dim)
        self.index.add(np.array(vectors).astype("float32"))
        logger.info("Index built successfully.")

        def search(self, query_vector, top_k=5):
        logger.info("Searching index...")
        D, I = self.index.search(np.array([query_vector]).astype("float32"),
top_k)
        logger.info(f"Search completed. Found {len(I[0])} results.")
        return [self.chunks[i] for i in I[0]]
```

Использование IndexFlatL2 обеспечивает точный поиск по L2-расстоянию, что критично для качества извлечения релевантных документов в небольших и средних корпусах.

Переранжирование

```
class Reranker:
        def __init__(self):
        logger.info("Loading Reranker model...")
        self.tokenizer =
AutoTokenizer.from_pretrained("cross-encoder/ms-marco-MiniLM-L-6-v2")
        self.model =
AutoModelForSequenceClassification.from_pretrained("cross-encoder/ms-ma
rco-MiniLM-L-6-v2")
        self.model.eval()
        logger.info("Reranker model loaded successfully.")

        def rerank(self, query, candidates):
        logger.info(f"Reranking candidates... ({len(candidates)} candidates)")
        inputs = self.tokenizer([f"{query} [SEP] {c}" for c in candidates],
return_tensors="pt", padding=True, truncation=True)
        scores = self.model(**inputs).logits.squeeze()
        sorted_indices = torch.argsort(scores, descending=True)
        logger.info("Reranking completed.")
        return [candidates[i] for i in sorted_indices]
```

Cross-encoder модель анализирует каждую пару запрос-документ целиком, что позволяет более точно оценить релевантность по сравнению с би-энкодерными подходами.

Формирование прогнозов

Модель ответа:

```
class ResponseProbJustification(BaseModel):
        probability: float
        justification: str
```

Функция создания промпта для генерации прогноза:

```
def create_probability_prompt_with_context(query, context):
        return [
        {
        "role": "system",
        "content": (
                "You are an expert forecaster. "
                "Given a binary event, estimate the probability (0-100%) that it
will happen, "
```

```
                "and provide a brief justification for your estimate. "
                "Respond strictly in the JSON format."
                f"The JSON object must use the schema: :
{json.dumps(ResponseProbJustification.model_json_schema(), indent=2)}"

        )
        },
        {
        "role": "user",
        "content": f"Event: {query}\n\n Context: {context}\n\n What is the
probability this event will occur? Please follow the required JSON format."
        }
        ]
```

Эксперименты и оценка качества

1. Описание датасета PROPHET

Для оценки эффективности разработанной системы прогнозирования событий использовался датасет PROPHET, представленный в работе Tao et al. (2025). Данный датасет состоит из вопросов, собранных с двух ведущих платформ краудсорсингового прогнозирования:

Metaculus — научно-ориентированная платформа для коллективного прогнозирования, где эксперты и энтузиасты делают предсказания о будущих событиях в области науки, технологий, политики и глобальных трендов. Платформа известна высоким качеством прогнозов благодаря строгим критериям разрешения вопросов и активному сообществу квалифицированных прогнозистов.

Manifold Markets — децентрализованная платформа прогнозирования, использующая механизмы предсказательных рынков. Пользователи могут создавать и торговать акциями событий, что позволяет агрегировать коллективное мнение о вероятности различных исходов через рыночные механизмы.

2. Структура датасета

Каждый элемент датасета PROPHET содержит следующие компоненты:

- **Вопрос** — бинарный вопрос о том, произойдет ли определенное событие (например, "Будет ли объявлена новая пандемия ВОЗ до конца 2024 года?")
- **Дата разрешения** — момент времени, когда стал известен фактический исход события
- **Итог события** — бинарный результат (произошло/не произошло)
- **Количество предсказаний** — число пользователей, принявших участие в прогнозировании данного события
- **Среднее предсказание сообщества** — агрегированная оценка вероятности события, полученная усреднением индивидуальных прогнозов пользователей
- **Набор релевантных статей** — коллекция новостных статей с высоким показателем CIL (Causal Intervened Likelihood), которые содержат информацию, существенно влияющую на корректность прогноза

Использование показателя CIL для отбора статей обеспечивает включение в датасет только тех источников информации, которые действительно способствуют принятию обоснованных решений о вероятности событий.

3. Экспериментальная установка

Формирование базы знаний

В ходе экспериментов использовались два подхода к формированию базы знаний для RAG-системы:

Локальный корпус (lc) — использование заранее собранного набора статей из датасета PROPHET, отобранных по критерию высокого CIL.

Динамический поиск через API Guardian (gd) — получение релевантных статей в режиме реального времени через API новостного агентства The Guardian на основе ключевых слов, извлеченных из вопроса.

В обоих случаях применялась строгая временная фильтрация новостных статей по дате публикации, чтобы предотвратить утечку данных (data leakage) — в базу знаний включались только статьи, опубликованные до даты постановки вопроса на платформах прогнозирования.

Архитектуры системы

Для сравнительного анализа были реализованы следующие конфигурации системы:

PlainLLM — базовая языковая модель без использования внешних источников информации, работающая исключительно на основе параметрических знаний.

NaiveRAG — классическая RAG-архитектура с последовательным поиском и генерацией, использующая либо локальный корпус (NaiveRAG_lc), либо динамический поиск (NaiveRAG_gd).

HybridRAG — усовершенствованная архитектура, комбинирующая множественные стратегии поиска и методы ранжирования контекста, также с двумя вариантами источников данных (HybridRAG_lc, HybridRAG_gd).

Метрики оценки

Для комплексной оценки качества системы использовались два типа задач:

Задача регрессии — предсказание вероятности события, где в качестве целевой переменной использовалось среднее предсказание сообщества (community prediction). Оценивались метрики MSE (Mean Squared Error), RMSE (Root Mean Squared Error) и MAE (Mean Absolute Error).

Задача классификации — предсказание бинарного исхода события (произойдет/не произойдет). Использовались метрики ROC-AUC, Precision, Recall, F1-score и Accuracy.

Дополнительно вычислялись **взвешенные метрики**, где вес каждого образца определялся количеством участников прогнозирования (total_predictors). Это позволяет придать больший вес образцам, где коллективная оценка вероятности события более надежна благодаря участию большего числа экспертов.

Обработка вариативности LLM

Для компенсации стохастической природы генерации языковых моделей каждое предсказание получалось трижды для каждого вопроса с последующим усреднением результатов. Это позволяет снизить влияние случайной вариативности ответов и получить более стабильные оценки производительности системы.

4. Результаты экспериментов

Регрессионные метрики

Результаты оценки качества предсказания вероятностей событий представлены в таблице 4.1.

Таблица 4.1. Регрессионные метрики качества прогнозирования

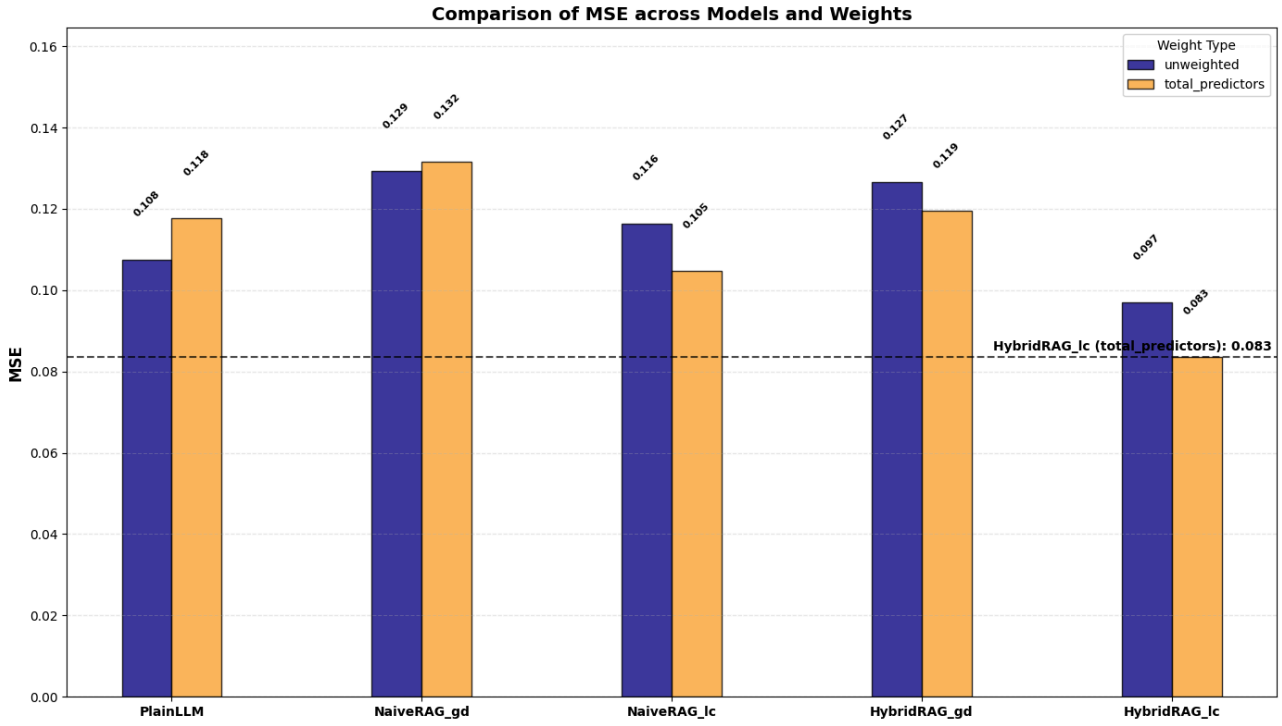
Модель	MSE	RMSE	MAE
PlainLLM	0.1075	0.3279	0.2566
NaiveRAG_gd	0.1293	0.3596	0.2806
NaiveRAG_lc	0.1164	0.3411	0.2678
HybridRAG_gd	0.1266	0.3558	0.2780
HybridRAG_lc	**0.0970**	**0.3114**	**0.2423**
Community	0.0000	0.0000	0.0000

Таблица 4.2. Взвешенные регрессионные метрики (вес: total_predictors)

Модель	MSE	RMSE	MAE
PlainLLM	0.1177	0.3430	0.2679
NaiveRAG_gd	0.1316	0.3628	0.2815
NaiveRAG_lc	0.1047	0.3236	0.2471
HybridRAG_gd	0.1195	0.3456	0.2722
HybridRAG_lc	**0.0835**	**0.2889**	**0.2211**
Community	0.0000	0.0000	0.0000

Наилучшие результаты демонстрирует архитектура HybridRAG с локальным корпусом статей, показывая улучшение по MSE на 9.8% по сравнению с базовой LLM и на 29% при использовании взвешенных метрик.

График 4.3. MSE



Классификационные метрики

Результаты бинарной классификации исходов событий представлены в таблицах 4.3 и 4.4.

Таблица 4.4. Классификационные метрики

Модель	ROC-AUC	Precision	Recall	F1	Accuracy
PlainLLM	0.4878	0.5000	0.3409	0.4054	0.5464
NaiveRAG_gd	0.6218	0.5625	0.6136	0.5870	0.6082
NaiveRAG_lc	0.6297	0.5472	0.6591	0.5979	0.5979
HybridRAG_gd	0.6237	0.5686	0.6591	0.6105	0.6186
HybridRAG_lc	**0.6642**	**0.5536**	**0.7045**	**0.6200**	**0.6082**
Community	0.8212	0.7442	0.7273	0.7356	0.7629

Таблица 4.5. Взвешенные классификационные метрики (вес: total_predictors)

Модель	ROC-AUC	Precision	Recall	F1	Accuracy
PlainLLM	0.4863	0.4700	0.2853	0.3551	0.5324
NaiveRAG_gd	0.6332	0.5311	0.6255	0.5744	0.5819
NaiveRAG_lc	0.6721	0.5786	0.6689	0.6205	0.6308
HybridRAG_gd	0.6584	0.5914	0.6689	0.6278	0.6421

HybridRAG_lc	**0.7137**	**0.5848**	**0.7135**	**0.6427**	**0.6421**
Community	0.8667	0.7722	0.7774	0.7748	0.7961

График 4.6. ROC-AUC

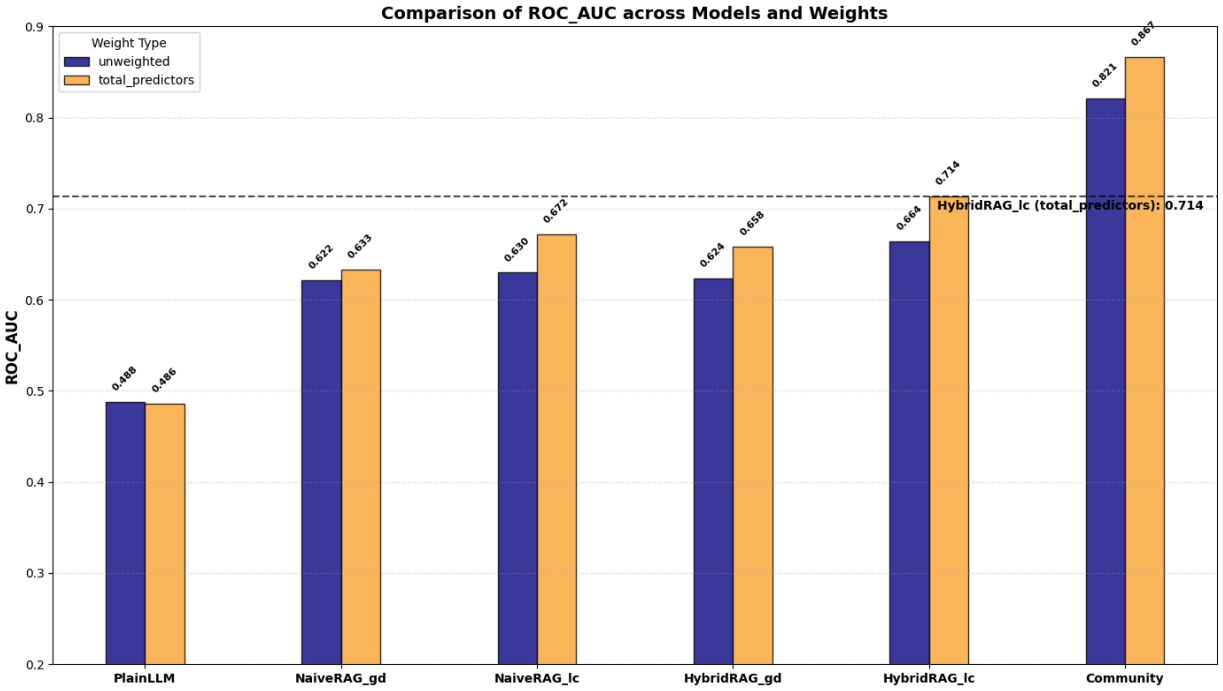
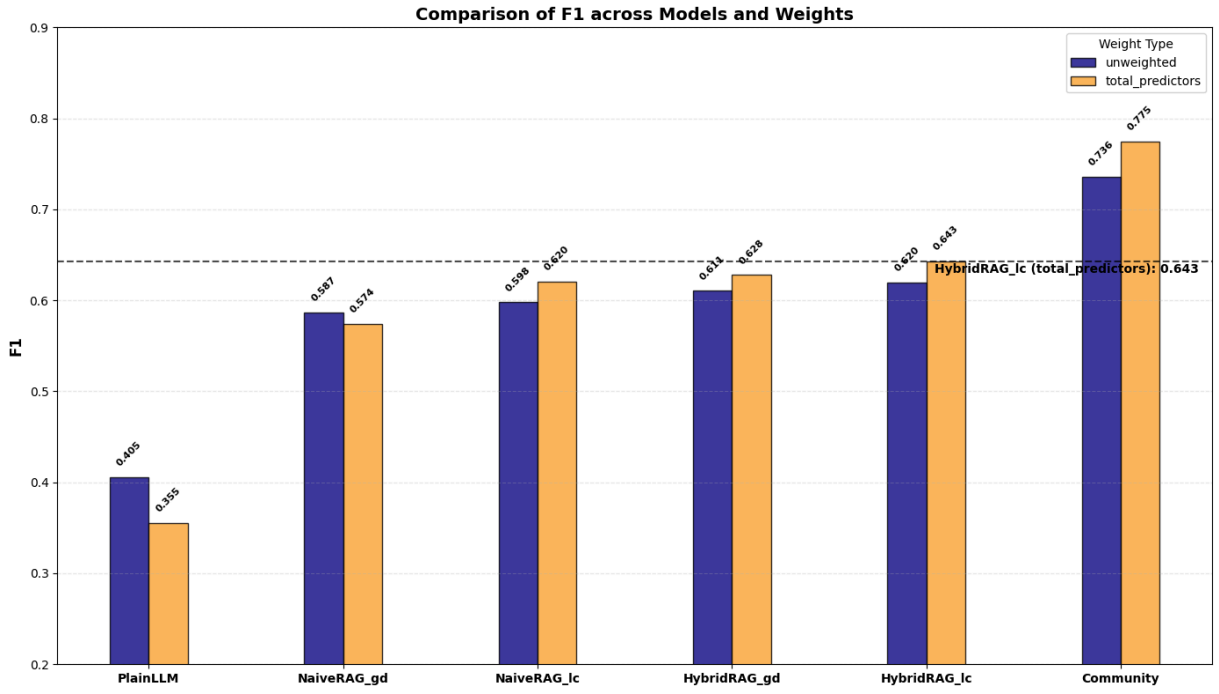


График 4.7. F1



5. Анализ результатов

Сравнительная эффективность архитектур

Результаты экспериментов демонстрируют следующие ключевые закономерности:

Превосходство RAG над базовой LLM: Все RAG-архитектуры показывают значительное улучшение качества прогнозирования по сравнению с чистой языковой моделью. Наиболее выраженное улучшение наблюдается в классификационных задачах, где ROC-AUC возрастает с 0.49 до 0.66-0.71.

Эффективность локального корпуса: Конфигурации с использованием предварительно отобранных статей (lc) стабильно превосходят варианты с динамическим поиском (gd). Это указывает на важность качественной курации источников информации и эффективность применения метрики CIL для отбора релевантных документов.

Преимущества гибридной архитектуры: HybridRAG демонстрирует наилучшие результаты благодаря комбинированию множественных стратегий поиска и улучшенным методам обработки контекста.

Паттерн высокого Recall при умеренном Precision: RAG-системы демонстрируют Recall на уровне 0.70-0.71, что сопоставимо с показателями сообщества (0.73-0.78). Однако Precision составляет лишь 0.55-0.58 против 0.74-0.77 у community predictions.

Интерпретация результатов: Высокий Recall указывает на способность системы корректно идентифицировать большинство событий, которые действительно произойдут. Относительно низкий Precision свидетельствует о склонности системы к ложноположительным предсказаниям — переоценке вероятности наступления событий.

Данный паттерн может объясняться особенностями обучения языковых моделей, которые могут проявлять оптимистическую предвзятость при интерпретации неоднозначной информации из новостных источников.

Эффект взвешивания

Применение взвешенных метрик, учитывающих количество участников прогнозирования, приводит к улучшению показателей всех RAG-систем. Это подтверждает гипотезу о том, что система показывает лучшие результаты на вопросах с большим экспертным консенсусом, где коллективные предсказания сообщества более надежны.

Сравнение с человеческими экспертами

Несмотря на значительные улучшения относительно базовых LLM, все автоматические системы существенно уступают коллективным предсказаниям сообщества. Community predictions достигают ROC-AUC 0.82-0.87, в то время как лучшая RAG-система показывает 0.66-0.71. Это указывает на наличие значительного пространства для дальнейших улучшений в области автоматического прогнозирования событий.

6. Ограничения и направления развития

Выявленные ограничения

Проблема переоценки рисков: Склонность к ложноположительным предсказаниям может ограничивать применимость системы в критически важных областях, где цена ошибочного прогнозирования высока.

Зависимость от качества источников: Значительное влияние метода формирования базы знаний на качество результатов подчеркивает важность развития более совершенных методов отбора и ранжирования релевантной информации.

Разрыв с экспертным сообществом: Существенное отставание от human-in-the-loop подходов указывает на необходимость интеграции человеческой экспертизы в процесс автоматического прогнозирования.

Перспективы улучшения

Результаты экспериментов указывают на следующие направления развития:

- Разработка более совершенных методов калибровки вероятностных оценок для снижения систематических смещений
- Интеграция мульти-агентных подходов и механизмов коллективного принятия решений
- Улучшение методов извлечения и синтеза информации из разнородных источников
- Развитие адаптивных механизмов обновления базы знаний в реальном времени

Выводы

В ходе выполнения данной работы была успешно разработана и протестирована система прогнозирования событий на основе RAG-архитектуры, которая демонстрирует значительные преимущества перед базовыми языковыми моделями.

Ключевые достижения:

1. **Архитектурное решение**: Реализована модульная RAG-система с поддержкой двух источников данных (API Guardian и датасет PROPHET) и трех конфигураций обработки (PlainLLM, NaiveRAG, HybridRAG).

2. **Повышение качества прогнозирования**: HybridRAG с локальным корпусом показал улучшение MSE на 9.8% по сравнению с базовой LLM и увеличение ROC-AUC с 0.49 до 0.66 в задачах классификации.

3. **Эффективность курации данных**: Использование предварительно отобранных статей с высоким CIL значительно превосходит динамический поиск, подтверждая важность качественной подготовки базы знаний.

Основные ограничения:

- Склонность к переоценке вероятности событий (высокий Recall при умеренном Precision)
- Существенное отставание от коллективных экспертных прогнозов (ROC-AUC 0.66 против 0.82-0.87)
- Зависимость качества результатов от метода формирования контекста

Практическая значимость:

Разработанная система может применяться как инструмент поддержки принятия решений в задачах краткосрочного прогнозирования, особенно в сценариях, где требуется быстрая обработка больших объемов текстовой информации. Модульная архитектура обеспечивает адаптируемость к различным предметным областям.

Список литературы:

[1] Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv preprint arXiv:2005.11401. https://arxiv.org/abs/2005.11401

[2] Gao, Y., Xiong, Y., Gao, X., et al. (2023). Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv preprint arXiv:2312.10997. https://arxiv.org/abs/2312.10997

[3] Chang, T., Li, Z., Wang, H., et al. (2024). A Comprehensive Evaluation of Large Language Models on Temporal Event Forecasting. arXiv preprint arXiv:2407.11638. https://arxiv.org/abs/2407.11638

[4] Yu, H.Q., McQuade, F. RAG-KG-IL: A Multi-Agent Hybrid Framework for Reducing Hallucinations and Enhancing LLM Reasoning through RAG and Incremental Knowledge Graph Learning Integration. arXiv preprint arXiv:2503.13514. https://www.arxiv.org/abs/2503.13514

[5] Tao, X., Wang, Y., Chen, L., et al. (2025). PROPHET: An Inferable Future Forecasting Benchmark with Causal Intervened Likelihood Estimation. arXiv preprint arXiv:2504.01509. https://www.arxiv.org/abs/2504.01509