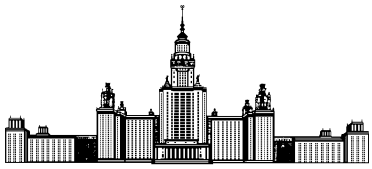

Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Левыкин Александр Михайлович

Детекция галлюцинаций больших языковых моделей

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
д.ф.-м.н., профессор
Воронцов К.В.

Москва, 2025

Содержание

Аннотация

В условиях растущих темпов использования больших языковых моделей как в повседневной жизни, так и в критически важных сферах, становится очевидной важность точности и достоверности ответов. В данной работе рассматривается задача детекции галлюцинаций языковых моделей с распознованием на уровне отдельных слов. В работе проводятся эксперименты с пятью моделями и предлагается новый процесс обработки ответов, использующий методы информационного поиска. Эксперименты на данных конкурса SemEval-2025 и собственном датасете, построенном на основе HaluEval с использованием GPT-4 для переразметки данных, показали высокую эффективность предложенного решения. Разработанная система легла в основу решения, обеспечившего победу в конкурсе SemEval-2025.

1 Введение

Быстрое развитие больших языковых моделей (LLM) приводит к значительному улучшению их способностей в генерации текста, ответах на вопросы и решении других задач обработки естественного языка (NLP) [23, 4]. Однако такие модели часто генерируют текст, содержащий недостоверные или ложные утверждения, известные как галлюцинации [13]. Эти ошибки могут возникать из-за недостатков в обучении моделей, ограниченности данных или сложности задачи согласования с реальными фактами [29]. Галлюцинации ограничивают применимость LLM в критически важных областях, где точность информации играет ключевую роль [22]. Создание систем доверенного искусственного интеллекта становится актуальной задачей, так как такие системы способствуют повышению надежности и доверия к языковым моделям, а также улучшению качества их применения в реальных сценариях [9].

Задача детекции галлюцинаций в больших языковых моделях является одной из подзадач обработки естественного языка, которая направлена на выявление ложных или недостоверных ответов, генерируемых моделями. Применение детекции галлюцинаций может быть полезно в следующих областях:

- В медицинских системах поддержки принятия решений идентификация ложных данных в ответах модели критически важна: ошибочные рекомендации по лечению, неточные диагнозы или недостоверные медицинские факты могут привести к серьезным последствиям для жизни и здоровья пациентов. Детекция галлюцинаций помогает снизить подобные риски, обеспечивая обоснованность ответов моделей, а также актуальность и достоверность источников, используемых при генерации [1, 22].

- В юридических консультационных системах и автоматизированном анализе документов недостоверная информация может привести к ошибочным выводам, что особенно важно при подготовке к судебным разбирательствам, анализе законодательства и составлении договоров. Методы детекции галлюцинаций позволяют

выявлять потенциально неверные интерпретации правовых норм, исключая влияние вымышленных или устаревших данных [7].

- В сфере финансового анализа и автоматизированного трейдинга ложные данные могут привести к ошибочным инвестиционным решениям и значительным финансовым потерям. Детекция галлюцинаций помогает проверять корректность финансовых отчетов, прогнозов и рекомендаций, предоставляемых языковыми моделями, снижая риски для пользователей таких систем [14, 26].

- В общедоступных вопрос-ответных системах и чат-ботах (ChatGPT, DeepSeek, Gemini) применение детекции галлюцинаций позволяет улучшить качество и достоверность предоставляемых ответов, помогая пользователям получать более точную информацию [8, 15].

- детекция галлюцинаций актуальна в таких задачах, как data-to-text generation [24] и суммаризация документов [5], где важно, чтобы модели не искажали исходные данные.

Галлюцинацией языковой модели называют ответ (или его часть), сгенерированный LLM, который не соответствует входным данным (промпту) или ранее сгенерированному контексту, либо противоречит общеизвестным фактам о мире. Задачи детекции галлюцинаций в больших языковых моделях можно классифицировать по нескольким основным признакам, включая подходы к детекции, использование эталонных данных и уровень детализации анализа.

В зависимости от наличия в задаче эталонного источника данных, методы детекции делятся на две группы:

- Детекция галлюцинаций с использованием источника (reference-based hallucination detection), при которой сгенерированный текст сравнивается с имеющимся эталонным источником данных. Этот подход успешно применяется в таких задачах, как суммаризация документов [5], машинный перевод [35], генерация текста на основе данных [24], создание подписей к изображениям и прочих задачах, где изначально задан некоторый достоверный источик, относительно которого идет поиск галлюцинаций и проверки фактов. Однако для многих задач генерации текста общего формата эталонные данные могут быть недоступны, что ограничивает применимость таких методов. Например, в диалоговых системах реального времени, таких как чат-боты, модель часто генерирует ответы без возможности сравнения с эталоном.

- Детекция галлюцинаций без использования источника (reference-free hallucination detection). В условиях, где эталонные данные отсутствуют, используются методы безэталонной детекции галлюцинаций, которые основываются только на контексте, правилах, встроенных в модель и информации, впитанной ею при обучении [17]. Такие подходы становятся актуальными в системах, работающих в реальном времени

(например, чат-боты [8, 15]), где сравнение с эталоном невозможно ввиду его отсутствия. Эти методы направлены на определение неконсистентных, несвязанных, отличных от запроса пользователя ответов и потенциально ложных утверждений исключительно по информации, доступной модели в контексте текущего сеанса взаимодействия.

В работе [30] предлагается общее деление галлюцинаций на внутренние и внешние, а в работе [36] - на галлюцинации, противоречащие запросу, либо контексту, либо фактам. Обобщая обе работы, предлагается следующая таксономия галлюцинаций:

- Внутренние (intrinsic) галлюцинации. Они в свою очередь делятся на:
  - Противоречащие вводу (input-conflict) галлюцинации — это случай, когда генерируемый текст не соответствует запросу пользователя.
  - Противоречащие контексту (context-conflict) галлюцинации - случай, когда модель генерирует высказывание, противоречащее ранее сгенерированной ею информации.
- Внешние (extrinsic) галлюцинации - определяются различно в зависимости от задачи:
  - генерируемый текст, не соответствующий общеизвестным фактам реального мира (для задач без источника),
  - генерируемый текст, который не может быть подтвержден информацией, содержащейся во входных данных - эталонном документе (для задач с источником).

Пример с введённой выше таксономией представлен на рис. 1.

Кроме того, подходы к решению задачи детекции галлюцинации можно поделить на два вида по степени детальности обнаружения:

- Классификация на уровне предложений или документов (sentence-level, document-level classification): Многие подходы анализируют галлюцинации на уровне предложения или всего документа [37]. Однако такой подход может быть недостаточно точным для выявления конкретных ложных утверждений внутри текста, так как он классифицирует весь текст в целом.
- Классификация на уровне токенов (token-level classification): В альтернативных методах детекция галлюцинаций осуществляется на уровне отдельных токенов [17, 30]. Такой подход позволяет более точно идентифицировать ложные утверждения в тексте и, в некоторых случаях, корректировать сгенерированный текст в реальном времени (например, управляя вероятностью появления тех или иных токенов). Этот метод более эффективен для задач, требующих высокую точность детекции и исправления. Кроме того, детекция на уровне токенов позволяет исправлять лишь

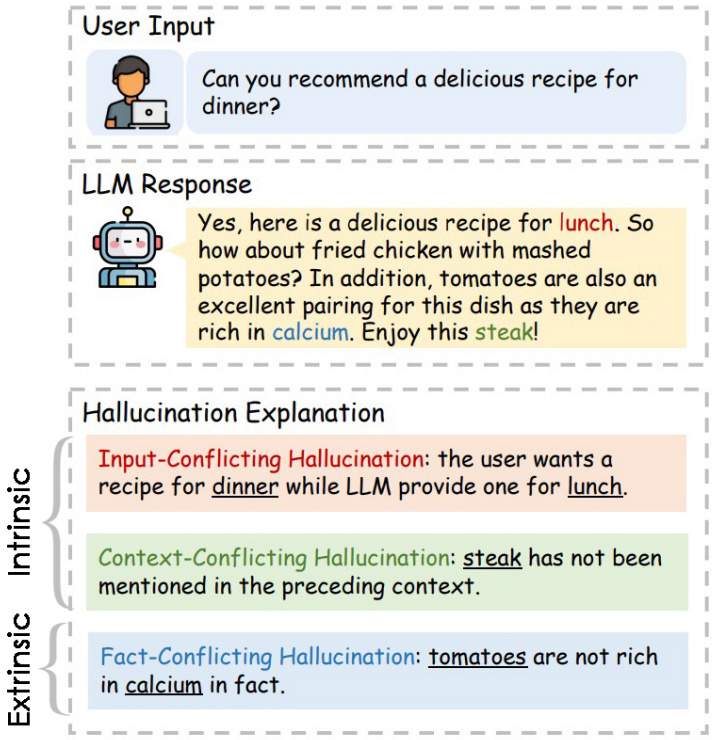

Рис. 1: Таксономия галлюцинаций на примере

небольшую часть ответа, а также позволяет подсвечивать пользователю именно те факты, в которых модель не уверена.

Задача детекции галлюцинаций без источника, с классификацией на уровне токенов, является наиболее сложным типом с той точки зрения, что в нем не задан документ-образец, относительно которого идёт поиск галлюцинаций. Кроме того, требование классификации каждого токена вынуждает применять более тонкие и выверенные подходы. Именно эта задача представляет наибольший интерес для исследований.

1.1 Цель работы

Целью данной работы является исследование различных методов детекции галлюцинаций в задаче без источника с классификацией на уровне токенов. Для достижения цели поставлены следующие задачи:

- Экспериментальное исследование рекуррентных моделей, генеративных моделей (LLM), дообучение моделей распознавания именованных сущностей (NER).
- Разработка нового процесса обработки ответов, основанного на методах информационного поиска, проведение экспериментов, подтверждающих его эффективность.

2 Постановка задачи

Пусть задано:

- Запрос пользователя (query, промпт, задача) — последовательность токенов $Q = \{q_1, \ldots, q_m\}$, где m — длина запроса.
- Ответ модели — последовательность токенов $X = \{x_1, x_2, \ldots, x_n\}$, где n — длина ответа.

Целью является детекция фрагментов ответа, являющихся галлюцинациями. Формально, задача состоит в нахождении множества непрерывных подстрок в ответе X, каждая из которых представляет собой галлюцинацию. Обозначим множество таких фрагментов

$$H = \{(i_1, j_1), \ldots, (i_K, j_K)\},$$

где $1 \leq i_k \leq j_k \leq n$, и подстрока $\overline{x_{i_k} x_{i_k+1} \ldots x_{j_k}}$ интерпретируется как галлюцинация.

При допущении отсутствия вложенных и пересекающихся фрагментов задача эквивалентна задаче классификации токенов на 3 класса (BIO-разметка: Begin, Inside, Out), где каждому токену x_i ставится в соответствие вектор меток $y_i \in \{0,1\}^3 : \sum_{j=1}^{3} y_{ij} = 1$, определяемый следующим образом:

$$y_i = \begin{cases} (1,0,0), & \text{если } x_i \text{ является началом галлюцинации,} \\ (0,1,0), & \text{если } x_i \text{ является продолжением галлюцинации,} \\ (0,0,1), & \text{если } x_i \text{ не относится к галлюцинации.} \end{cases}$$

Обучающая выборка представляется как множество размеченных объектов:

$$\mathcal{D}_{\text{train}} = \left\{ (Q^{(k)}, X^{(k)}, Y^{(k)}) \right\}_{k=1}^{N},$$

где $Q^{(k)}$ — запрос, $X^{(k)} = \{x_1^{(k)}, \ldots, x_{n_k}^{(k)}\}$ — ответ модели, и $Y^{(k)} = \{y_1^{(k)}, \ldots, y_{n_k}^{(k)}\}$, $y_i^{(k)} \in \{0,1\}^3 : \sum_{j=1}^{3} y_{ij}^{(k)} = 1$.

Требуется построить модель f_θ, параметризованную вектором параметров θ, которая по входной паре (Q, X) предсказывает последовательность меток:

$$\hat{Y} = f_\theta(Q, X) = \{\hat{y}_1, \ldots, \hat{y}_n\}, \quad \hat{y}_i \in [0,1]^3 : \sum_{j=1}^{3} \hat{y}_{ij} = 1.$$

Построение модели производится на $\mathcal{D}_{\text{train}}$, а её качество оценивается на отложенной выборке $\mathcal{D}_{\text{test}}$ с использованием метрик F1 Macro и IoU (Intersection over Union).

3 Модели и методы

3.1 Рекуррентные модели

Для случаев, когда помимо текста ответа $X = \{x_1, \ldots, x_n\}$ доступны логиты модели $L = \{l_1, \ldots, l_n\}$, можно обучить модель f_θ, классифицирующую токены на основе только

последовательности логитов.

Архитектура состоит из рекуррентной модели (например, LSTM [12]), за которой следует линейный слой, понижающий размерность до 3 (по числу классов), и слой softmax, преобразующий выходы в вероятности:

$$\hat{Y} = f_\theta(L) = \{\hat{y}_1, \ldots, \hat{y}_n\}, \quad \hat{y}_i \in [0,1]^3, \quad \sum_{j=1}^{3} \hat{y}_{ij} = 1.$$

Истинные метки $Y = \{y_1, \ldots, y_n\}$ представлены в one-hot формате, $y_i \in \{0,1\}^3$: $\sum_{j=1}^{3} y_{ij} = 1$.

Обучение модели производится на обучающей выборке $\mathcal{D}_{\text{train}} = \{(L^{(k)}, Y^{(k)})\}_{k=1}^{N}$ путём минимизации средней функции потерь кросс-энтропии:

$$\mathcal{L}(\mathcal{D}_{\text{train}}, \theta) = \frac{1}{N} \sum_{k=1}^{N} \text{CE}(Y^{(k)}, f_\theta(L^{(k)})) = -\frac{1}{N} \sum_{k=1}^{N} \sum_{i=1}^{n_k} \sum_{j=1}^{3} y_{ij}^{(k)} \log \hat{y}_{ij}^{(k)} \longrightarrow \min_\theta, \qquad (1)$$

где $\hat{y}_{ij}^{(k)}$ — вероятность класса j для i-го токена в k-м примере.

Подход на основе рекуррентных моделей, использующих только логиты, позволяет выявлять временные зависимости без опоры на лексическое содержание, снижая влияние языковых особенностей и фокусируясь на статистических закономерностях. Кроме того, сами логиты могут нести информацию о степени уверенности модели, что логично связывается с возникновением галлюцинаций. Однако такой подход не учитывает исходный запрос пользователя и семантику токенов, что существенно ограничивает точность классификации.

3.2 Модели распознавания именованных сущностей

Другим подходом к детекции галлюцинаций является формулировка задачи как задачи распознавания именованных сущностей (Named Entity Recognition, NER) — последовательной классификации токенов с помощью предобученных трансформеров. Этот подход опирается на использование архитектур, состоящих из энкодера и классификатора и позволяет учитывать как исходный запрос пользователя, так и лексико-семантические особенности сгенерированного ответа.

Модель состоит из двух основных компонентов:

- Энкодер (обычно используется предобученный трансформер, например, BERT), который принимает на вход последовательность токенов, объединяющую (например, через [SEP]-токен) запрос пользователя $Q = \{q_1, ..., q_m\}$ и ответ модели $X = \{x_1, ..., x_n\}$. На выходе из энкодера имеем для каждого токена ответа x_i его векторное представление e_i, в которое заложена информация о словах, окружающих данное слово, и о вопросе. Схема представлена на рис. 2.

- Классификатор — полносвязный линейный слой, принимающий эмбеддинги токе-

нов из энкодера и возвращающий вероятности принадлежности каждого токена ответа x_i к одному из трёх классов.

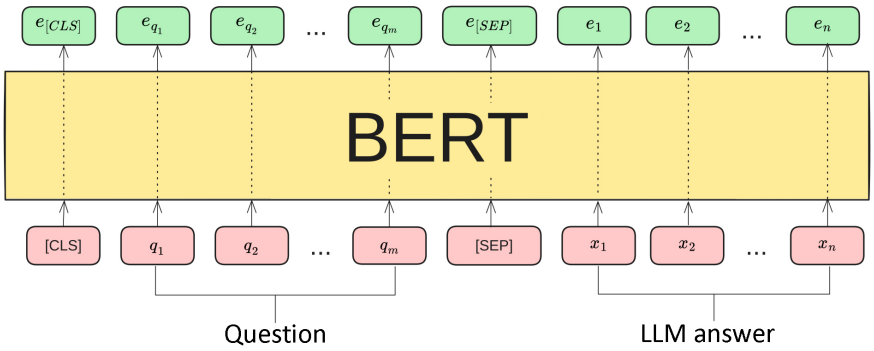

Рис. 2: Получение векторных представлений токенов с помощью энкодера

Пусть $E \in \mathbb{R}^{n \times d}$ — матрица скрытых представлений (эмбеддингов) размерности d, соответствующих токенам ответа $X = \{x_1, \ldots, x_n\}$, полученных после пропуска всей последовательности $[Q; X]$ через энкодер трансформера. К каждому вектору e_i применяется линейный классификатор, выдающий распределение по трём меткам:

$$\hat{y}_i = \text{softmax}(We_i + b), \quad i = 1, \ldots, n, \quad \hat{y}_i \in [0,1]^3$$

где $W \in \mathbb{R}^{3 \times d}$, $b \in \mathbb{R}^3$ — обучаемые параметры модели.

В процессе обучения, при наличии обучающей выборки $\mathcal{D}_{\text{train}} = \left\{(Q^{(k)}, X^{(k)}, Y^{(k)})\right\}_{k=1}^{N}$, аналогично подходу с рекуррентной моделью, используется функция потерь на основе кросс-энтропии между истинными метками $Y^{(k)} \in \{0,1\}^{n_k \times 3}$ и предсказаниями $\hat{Y}^{(k)} \in [0,1]^{n_k \times 3}$:

$$\mathcal{L}(\mathcal{D}_{\text{train}}, \theta) = \frac{1}{N} \sum_{k=1}^{N} \text{CE}(Y^{(k)}, f_\theta(L^{(k)})) = -\frac{1}{N} \sum_{k=1}^{N} \sum_{i=1}^{n_k} \sum_{j=1}^{3} y_{ij}^{(k)} \log \hat{y}_{ij}^{(k)} \longrightarrow \min_\theta .$$

Подход, основанный на трансформерных энкодерах, обладает рядом преимуществ. Во-первых, модель получает на вход как пользовательский запрос Q, так и сгенерированный ответ X, что позволяет анализировать их взаимосвязь и выявлять возможные несоответствия. Во-вторых, благодаря архитектуре трансформеров, модель способна учитывать сложные семантические и синтаксические зависимости, обеспечивая более точное понимание контекста. Кроме того, подход является более универсальным и может быть применён к любым языковым моделям, поскольку опирается исключительно на сгенерированный текст.

Из недостатоков стоит отметить отсутствие информации о логитах, что не позволяет явно учитывать степень «уверенности» модели в процессе генерации, хотя именно это может быть важным индикатором галлюцинации. Также трансформерные энкодеры требуют значительных вычислительных ресурсов, особенно при работе с длинными последовательностями.

Модели, основанные на токен-классификации, в частности подходы распознавания именованных сущностей (NER), успешно применялись для детекции галлюцинаций в текстах, сгенерированных LLM. В работе [2] предложен метод выявления галлюцинаций путём извлечения именованных сущностей и сопоставления их с эталонным контекстом. Такой способ особенно эффективен для обнаружения вымышленных имён, мест и чисел.

В [27] NER модель используется в составе многоуровневой системы детекции, где сущности проверяются на наличие в исходных документах. Совмещение NER с другими подходами, такими как NLI и анализ спанов, повышает точность и снижает количество ложных срабатываний.

Также в бенчмарке HaDes [10] представлена разметка галлюцинаций на уровне токенов, где среди прочего рассматриваются подходы, основанные на извлечении сущностей, как способ повышения интерпретируемости и локализации недостоверных фрагментов.

Таким образом, подход, основанный на NER, является мощным инструментом, особенно в ситуациях, когда важен лексический и семантический контекст между вопросом и ответом.

3.3 Генеративные языковые модели

В предложенном подходе для детекции галлюцинаций используется метод подбора инструкции (instruction-based метод), при которой большая языковая модель (LLM) генерирует текст, явно указывая на галлюцинации в ответе. В данном эксперименте вместо вывода вероятностей или меток на уровне токенов модель на естественном языке выделяет фрагменты текста, которые, по её мнению, являются галлюцинациями.

Одним из ключевых свойств крупных языковых моделей является способность решать задачи, на которые они не были специально обучены, используя инструкции (промпты), формирующие поведение модели в ходе генерации текста. Исследования по подбору промпта широко развились [25, 16] и получили термин “prompt engineering”, а способность генеративных моделей проявлять умения, которые от них не требовались при обучении, получила название эмерджентным поведением (emergent behavior) и повлекло множество исследований [31, 21].

Подбор эффективной инструкции осуществляется на обучающей выборке $\mathcal{D}_{\text{train}} = \left\{(Q^{(k)}, X^{(k)}, Y^{(k)})\right\}_{k=1}^{N}$. Цель состоит в формулировке промпта P, который бы направлял генеративную модель на явное выявление и маркировку токенов, содержащих недостоверную информацию. Эффективность сформированного промпта оценивается на основе критериев качества (F1 Macro, IoU) на $\mathcal{D}_{\text{train}}$.

Одним из ключевых достоинств Instruction-based подхода является высокая адаптивность: благодаря использованию текстовых инструкций, крупные языковые модели могут решать задачи, на которых они не были напрямую обучены. Это существенно расширяет возможности применения таких моделей без необходимости дополнительного обучения, что позволяет обходиться без сбора дополнительных размеченных данных и дообучения моделей, что особенно важно при ограниченных временных и финансовых

ресурсах. Отдельно стоит отметить интерпретируемость — генеративные модели при необходимости способны пояснять свой выбор.

Тем не менее, instruction-based подход имеет и ограничения. Наиболее существенным из них является чувствительность к формулировке промпта. Незначительные изменения в инструкции могут приводить к существенным изменениям в результатах, что требует ручной настройки и экспериментов. Также метод не предоставляет вероятностных оценок предсказаний, из-за чего трудно оценить уверенность модели в каждом конкретном ответе. Наконец, эффективность метода напрямую зависит от масштаба языковой модели: более компактные модели демонстрируют значительно худшие результаты, тогда как крупные модели требуют существенных вычислительных ресурсов, что увеличивает стоимость их применения в прикладных системах.

3.4 Подход Retrieval-Augmented Generation

Одной из ключевых причин возникновения галлюцинаций в больших языковых моделях является их ограничение обучающим корпусом: модели генерируют текст, опираясь на знания, зафиксированные в весах модели на момент обучения. Это делает их уязвимыми к устаревшей, неполной или просто отсутствующей информации. Подход Retrieval-Augmented Generation (RAG) предлагает решение этой проблемы за счёт интеграции механизмов информационного поиска в процесс генерации. RAG совмещает два компонента:

- Модуль извлечения информации (retriever) — получает релевантные документы из внешней базы знаний (например, Wikipedia, научные статьи, корпоративные базы данных) в ответ на заданный запрос;
- Модуль генерации (generator) — использует retrieved-документы в качестве дополнительного контекста для генерации ответа. В большинстве случаев используются генеративные LLM.

Пусть имеется коллекция документов $\mathcal{D} = \{d_1, d_2, \ldots, d_M\}$. В процессе формирования базы данных каждый документ d_i преобразуется в векторное представление с помощью предобученного энкодера (как правило, основанные на BERT архитектуре модели):

$$\mathbf{v}_i = \text{Encoder}(d_i), \quad \mathbf{v}_i \in \mathbb{R}^n$$

Вектор $\mathbf{v}_i$ представляет семантическое содержание документа d_i в n-мерном пространстве. Эти векторы сохраняются в векторной базе данных (например, Milvus или Qdrant), которая позволяет выполнять эффективный поиск ближайших соседей.

При поступлении запроса пользователя, запрос q кодируется тем же энкодером:

$$\mathbf{v}_q = \text{Encoder}(q), \quad \mathbf{v}_q \in \mathbb{R}^n$$

Затем производится поиск k наиболее близких документов $\{d_{(1)}, d_{(2)}, \ldots, d_{(k)}\} \subseteq \mathcal{D}$ по метрике сходства. Наиболее распростанена метрика косинусного сходства:

$$\text{sim}(q, d_i) = \frac{\mathbf{v}_q \cdot \mathbf{v}_i}{\|\mathbf{v}_q\| \cdot \|\mathbf{v}_i\|}$$

Могут использоваться и другие метрики, например, евклидово расстояние:

$$\text{dist}(q, d_i) = \|\mathbf{v}_q - \mathbf{v}_i\|_2$$

Наконец, модуль генерации (генеративная LLM) получает на вход исходный запрос q вместе с найденными документами $\{d_{(1)}, \ldots, d_{(k)}\}$. Ответ модели y генерируется с учётом этого внешнего контекста:

$$P(y \mid q, \{d_{(j)}\}) = \text{Generator}(q, d_{(1)}, \ldots, d_{(k)})$$

Таким образом, вместо того чтобы полагаться исключительно на "внутренние знания" модели, RAG динамически подключает внешние источники, обогащая генерацию контекстной и проверенной информацией. Кроме того, подход позволяет повышать интерпретируемость, поскольку можно отследить, какие документы использовались, а также обеспечивать актуальность данных, особенно при использовании живых источников (например, web search).

В контексте детекции галлюцинаций RAG может использоваться и как основа для генерации, которая изначально предотвращает появление ложных утверждений, и как вспомогательный механизм верификации уже сгенерированного текста (как в данной работе) путём проверки наличия подтверждающей информации в retrieved-документах.

Подход Retrieval-Augmented Generation активно используется научным сообществом для повышения достоверности генераций и выявления галлюцинаций. Так, в работе [11] предложена метрика RagScore, которая измеряет согласованность ответа модели с retrieved-документами, позволяя оценивать степень галлюцинации без необходимости использования эталонных ответов. В исследовании [28] представлен FactScore — метод детекции галлюцинаций путём сопоставления фактов из ответа модели с релевантными документами из retrieval-модуля. Кроме того, в [20] приведён обзор различных стратегий, включая методы сравнения ответов с источниками, reranking retrieved-документов и выделения фрагментов, не подтверждённых знаниями, что позволяет использовать RAG не только как способ генерации, но и как основу для верификации сгенерированного вывода.

3.5 Автоматический подбор примеров

Few-Shot Learning (FSL) — это подход автоматического подбора небольшого количества примеров для генеративной модели. В контексте больших языковых моделей (LLM) данный подход реализуется без дополнительного дообучения модели, а за счёт

использования так называемого *in-context learning* [3, 19]: в качестве входа подаётся промт, включающий несколько примеров с соответствующими ответами.

Если x — целевой запрос, к которому необходимо сгенерировать ответ, и пусть $\mathcal{D}_{\text{support}} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$ — множество k примеров. Тогда входной промт, подаваемый в языковую модель, обогащается примерами из $\mathcal{D}_{\text{support}}$, и LLM принимает этот промт и генерирует предсказание $\hat{y}$, соответствующее ответу на x, опираясь на примеры из $\mathcal{D}_{\text{support}}$:

$$P(y \mid x, \mathcal{D}_{\text{support}}) = \text{Generator}(x, \quad (x_1, y_1), \ldots, (x_k, y_k))$$

Преимущество подхода состоит в том, что он позволяет использовать LLM без дополнительного дообучения, что критично при отсутствии доступа к внутренним параметрам модели, требует минимального количества размеченных данных, а также хорошо адаптируется к новым задачам или доменам.

Качество генерации напрямую зависит от метода формирования множества примеров $\mathcal{D}_{\text{support}}$. Были предложены различные стратегии выбора:

1. **Similarity-Based Retrieval** [18]: отбор наиболее схожих с текущим запросом примеров с использованием векторных представлений (например, BERT embeddings). Используется косинусное расстояние:

$$\text{sim}(x, x_i) = \frac{\langle \text{emb}(x), \text{emb}(x_i) \rangle}{\|\text{emb}(x)\| \cdot \|\text{emb}(x_i)\|}$$

2. **Diverse Sampling** [6]: выбор максимально разнообразных примеров, покрывающих разные аспекты задачи.

3. **Hard Example Mining** [6]: выбор примеров, на которых модель ранее ошибалась, для фокусировки на сложных случаях.

4. **Prototype-Based Selection** [32]: генерация прототипов классов и выбор ближайших к ним примеров.

5. **Uncertainty Sampling** [34]: использование примеров, к которым модель проявляет наибольшую неуверенность (например, максимальная энтропия предсказания).

3.6 Предложенный метод

В данной работе предлагается многоэтапный процесс обработки (конвейер) детекции галлюцинаций в текстах, основанный на последовательном применении нескольких техник: больших языковых моделей (LLM), информационного поиска методом Retrieval-Augmented Generation, автоматического подбора примеров (Few-Shot Learning). Рассмотрим последовательно работу процесса.

Используем обозначения: Q - запрос пользователя, X - ответ модели, $\mathfrak{D}$ - коллекция документов, $\mathcal{D}_{\text{train}} = \left\{(Q^{(k)}, X^{(k)}, Y^{(k)})\right\}_{k=1}^{N}$ - обучающая выборка. Тогда предложенный метод можно представить следующим образом:

1. Получение набора релевантных документов для запроса Q:

$$\mathfrak{D}_Q = \{D_i\}_{i=1}^{M} = \text{Retrieve}(Q)$$

2. Формирование набора примеров $\mathcal{D}_{\text{support}}$ из обучающей выборки $\mathcal{D}_{\text{train}}$ для запроса Q:

$$\mathcal{D}_{\text{support}} = \left\{(Q^{(k)}, X^{(k)}, Y^{(k)})\right\}_{k=1}^{M} = f(Q, \mathcal{D}_{\text{train}})$$

3. Получение набора релевантных документов для каждого примера из $\mathcal{D}_{\text{support}}$:

$$\mathfrak{D}_{\text{support}} = \{\mathfrak{D}_{Q^{(1)}}, \ldots, \mathfrak{D}_{Q^{(M)}}\}, \text{где} \quad \mathfrak{D}_{Q^{(k)}} = \{D_i^{(k)}\}_{i=1}^{M} = \text{Retrieve}(Q^{(k)})$$

4. Формирование итогового промпта для генератора:

$$\text{Prompt} = \text{Concat}(Q, X, \mathfrak{D}_Q, \mathcal{D}_{\text{support}}, \mathfrak{D}_{\text{support}})$$

5. Получение обнаруженных галлюцинаций от генеративной модели:

$$H = \text{Generator}(\text{Prompt})$$

Предложенный подход комбинирует преимущества описанных ранее методов: мощь, гибкость LLM и фактологическую точность RAG. Кроме того, метод использует Few-Shot Learning для повышения адаптивности модели.

3.7 Методика оценивания модели

В качестве критериев качества в работе используются метрики F1 Macro и IoU (Intersection over Union). Опишем, как они вычисляются для одного объекта, а для всего датасета значение получится усреднением. В соответствие введённым при постановке задачи обозначениям, пусть Q — запрос, $X = \{x_1, \ldots, x_n\}$ — ответ модели, и $Y = \{y_1, \ldots, y_n\}$, $\hat{Y} = \{\hat{y}_1, \ldots, \hat{y}_n\}$ - истинные метки и предсказания модели соответственно, где $y_i, \hat{y}_i \in 1, 2, 3$. Здесь для простоты формул сделан переход от one-hot представления меток вектором рамерности 3: $y_i \in \{0, 1\}^3$ к эквивалентному скалярному представлению, принимающему одно из 3-х значений: $y_i \in \{1, 2, 3\}$.

True Positive (TP_i) - количество токенов, которые модель правильно классифицировала как принадлежащие классу i:

$$TP_i = \sum_{j=1}^{n} \mathbb{I}(\hat{y}_j = i \wedge y_j = i)$$

где $\mathbb{I}(\cdot)$ — индикаторная функция, принимающая значение 1, если условие выполняется, и 0 в противном случае.

False Positive (FP_i) - количество токенов, которые модель ошибочно классифицировала как принадлежащие классу i:

$$FP_i = \sum_{j=1}^{N} \mathbb{I}(\hat{y}_j = i \wedge y_j \neq i)$$

False Negative (FN_i) - количество токенов или фрагментов, которые модель ошибочно не отнесла к классу i:

$$FN_i = \sum_{j=1}^{N} \mathbb{I}(\hat{y}_j \neq i \wedge y_j = i)$$

Точность (Precision) для i-го класса показывает, какая доля токенов, предсказанных моделью как относящиеся к классу i, действительно относятся к этому классу:

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i}$$

Полнота (Recall) для i-го класса показывает, какую долю токенов класса i модель правильно предсказала, по отношению к общему числу токенов данного класса в данных:

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i}$$

F1-score для i-го класса является гармоническим средним точности и полноты и даёт общую оценку качества предсказания для класса i:

$$\text{F1}_i = 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

F1-macro (макроусреднённый F1-score) вычисляется как среднее значение F1-score по всем классам, где каждый класс имеет одинаковый вес:

$$\text{F1-macro} = \frac{1}{C} \sum_{i=1}^{C} \text{F1}_i$$

где C — количество классов. В рассматриваемой задаче $C = 3$.

Если свести задачу до бинарной классификации токенов, где токенам начала и продолжения будет соответствовать метка 1, а токенам вне галлюцинаций - метка 0, то можно рассчитать значение метрики Intersection over Union как:

$$\text{IoU} = \frac{TP}{TP + FP + FN}$$

Vожно показать, что в случае бинарной классификации IoU и F1-score связаны следующим образом:

$$\text{IoU} = \frac{\text{F1}}{2 - \text{F1}} \quad \text{и, наоборот,} \quad \text{F1} = \frac{2 \cdot \text{IoU}}{1 + \text{IoU}}$$

4 Эксперименты

4.1 Данные

4.1.1 SemEval-2025

В рамках соревнования *SemEval-2025* [1] был предоставлен датасет, размеченный по задаче детекции фактологических галлюцинаций. Данные разделены на валидационную и тестовую выборки.

Валидационная часть состоит из подвыборок на 10 различных языках, каждая из которых содержит по 50 объектов. Тестовая часть включает подвыборки на 14 языках, в каждой из которых — по 150 объектов.

Каждый объект датасета представляет собой словарную структуру с полями, описывающими вход модели, её ответ, вероятности галлюцинации для токенов и другую вспомогательную информацию. Ниже приведён пример одного из объектов:

Листинг 1: Пример объекта валидационной выборки

```
{
  "lang": "EN",
  "model_input": "In which city was David Sandberg born?",
  "model_output_text": "David Sandburg was born in Stockholm, Sweden
     .",
  "model_id": "tiiuae/falcon-7b-instruct",
  "soft_labels": [
    {"start": 27, "prob": 0.909, "end": 36},
    {"start": 36, "prob": 0.091, "end": 44}
  ],
  "hard_labels": [[27, 36]],
  "model_output_tokens": [
    "David", " Sand", "burg", " was", " born", " in",
    " Stockholm", ",", " Sweden", ".", "<|endoftext|>"
  ],
  "model_output_logits": [
    -5.99, -14.99, -11.57, -12.78, -7.70,
    -9.53, -4.74, -8.40, -9.93, -13.37, -8.34
  ]
}
```

[1] https://helsinki-nlp.github.io/shroom/

Поле `model_input` содержит входной запрос, а `model_output_text` — сгенерированный моделью текст, `model_id` - название модели, сгенерировавшей галлюцинацию. Массив `soft_labels` отражает вероятности того, что соответствующий токен является галлюцинацией. `hard_labels` — бинарная аннотация, указывающая на позицию токенов, однозначно помеченных как галлюцинации. Также для каждого токена доступны его логиты (`model_output_logits`) и текстовое представление (`model_output_tokens`). Каждый объект размечался 10 разметчиками, затем их разметки усреднялись, так получались soft_labels. Инструкция для разметчиков приведена в приложении А. Дополнительная ценность датасета в том, что здесь представлены реальные галлюцинации языковых моделей (а не искусственно созданные, путем запроса об этом в промпте). Для ответа на каждый вопрос авторы запускали несколько моделей и конфигураций, тем самым повышая вероятность галлюцинации. Намеренно использовались не самые большие языковые модели, т.к. вероятность их галлюцинаций куда выше. Датасет содержит широкий спектр тем: биология, история, спорт, технологии, география, литература и другие.

Такая структура позволяет как обучать модели в формате sequence labeling, так и проводить подробный анализ ошибок и уверенности модели в своих предсказаниях.

4.1.2 HaluEval

Ввиду ограниченного объёма датасета SemEval возникла необходимость поиска более крупного корпуса с фактологическими галлюцинациями. Было установлено, что в открытом доступе отсутствуют датасеты с аналогичным форматом задачи — классификацией токенов.

Наиболее релевантным по тематике оказался датасет HaluEval [15] [2]. Его раздел QA Data содержит 10 тысяч объектов вида (знание [knowledge], вопрос, правильный ответ, ответ модели). Пример исходного объекта из оригинального датасета HaluEval приведён ниже:

Листинг 2: Исходный пример из HaluEval

```
{
  "knowledge": "Arthur's Magazine (1844   1846) was an American
     literary periodical published in Philadelphia in the 19th
     century. First for Women is a woman's magazine published by
     Bauer Media Group in the USA.",
  "question": "Which magazine was started first Arthur's Magazine or
     First for Women?",
  "right_answer": "Arthur's Magazine",
  "hallucinated_answer": "First for Women was started first."
}
```

[2] https://github.com/RUCAIBox/HaluEval

При построении раздела QA Data HaluEval авторы взяли 10000 случайных объектов-троек (знание, вопрос, правильный ответ) из датасета HotpotQA [33]. Датасет HotpotQA содержит 112,779 объектов (троек). Вопросы для HotpotQA были написаны разметчиками, которым подавались два абзаца статей из Wikipedia (был взят весь корпус статей на английском языке), и задачей размечика было написать вопрос, ответ на который требует знания обоих абзацев. После написания вопроса разметчик сразу оставлял и правильный ответ. Разметка происходила на платформе Amazon Mechanical Turk. Так как был взят весь корпус Wikipedia, Датасет содержит широкий спектр тем: фильмы, бизнес, музыка, история, здоровье, языки, технологии, спорт, география, литература и другие. Структуры предложений представлены на диаграмме 3. Виды ответов представлены в таблице 1.

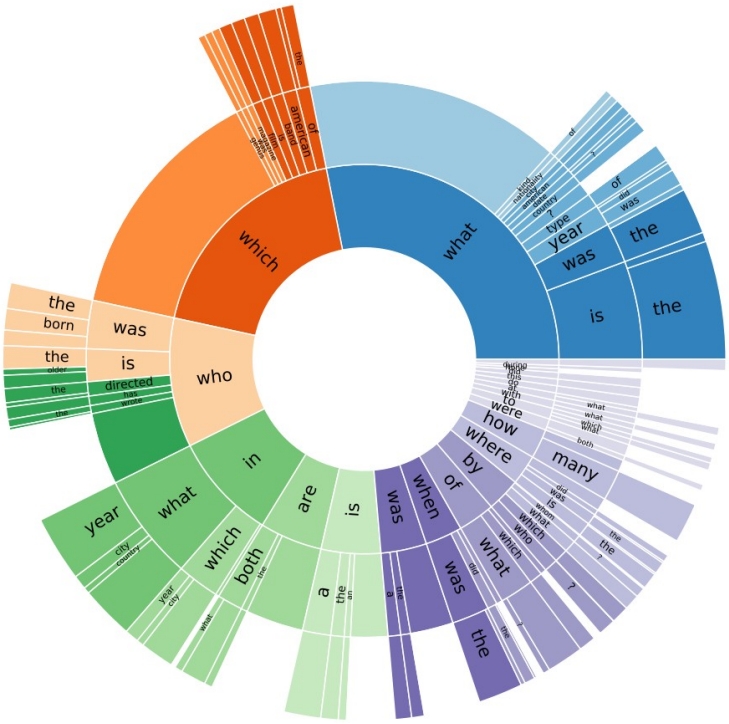

Рис. 3: Структуры вопросов в датасете HotpotQA.

Для генерации ответа с галлюцинацией авторы HaluEval взяли ChatGPT и ставили ей задачу ошибочно ответить на поставленный вопрос. Авторы использовали подходы Zero-Shot и Few-Shot. Затем из двух ответов отбирали тот, который GPT-судья назовёт правильным. Таким образом оставался наиболее правдоподобный ошибочный ответ. Примеры запросов для генерации и отбора приведены в приложении A=. Затем разметчики вручную валидировали, есть ли галлюцинация в ответе ChatGPT. Валидация происходила на половине датасета. Для разнообразия датасета было оставлено 100 объектов (1% от датасета), не содержащих галлюцинаций. Коэффициент каппа-статистики Коэна составил 0.811, что выражает очень высокий уровень согласия.

Датасет HaluEval демонстрирует высокое качество и релевантность данных, однако его структура не соответствует задаче классификации токенов: каждый пример

Тип ответа	%	**Пример**
Person	30	King Edward II, Rihanna
Group / Org	13	Cartoonito, Apalachee
Location	10	Fort Richardson, California
Date	9	10th or even 13th century
Number	8	79.92 million, 17
Artwork	8	Die schweigsame Frau
Yes/No	6	-
Adjective	4	conservative
Event	1	Prix Benois de la Danse
Other proper noun	6	Cold War, Laban Movement Analysis
Common noun	5	comedy, both men and women

Таблица 1: Типы ответов в HotpotQA.

представляет собой вопрос, правильный ответ и ответ модели, а разметка фрагментов, являющихся галлюцинациями модели, отсутствует.

В связи с этим было принято решение использовать автоматическое переформатирование датасета с помощью языковой модели GPT-4. Цель переразметки — получить точечные аннотации галлюциногенных фрагментов на уровне токенов. В результате была сформирована новая версия датасета, содержащая 10 000 размеченных объектов, каждый из которых представляет собой JSON-структуру с указанием границ галлюцинаций в выходном тексте.

После переразметки с помощью GPT-4 объект преобразуется в следующий формат, пригодный для задачи токен-классификации:

Листинг 3: Пример после переразметки GPT-4

```
{
  "knowledge": "Arthur's Magazine (1844   1846) was an American
     literary periodical published in Philadelphia in the 19th
     century. First for Women is a woman's magazine published by
     Bauer Media Group in the USA.",
  "model_input": "Which magazine was started first Arthur's Magazine
     or First for Women?",
  "right_answer": "Arthur's Magazine",
  "model_output": "First for Women was started first.",
  "hard_labels": [[0, 13]]
}
```

Для выполнения переразметки использовался специализированный промпт, направляющий модель на выявление фактологических несоответствий в сгенерированном ответе. Описание промпта приведено в приложении А.

Данная задача проста для модели, т.к. модель имеет непосредственный доступ к правильному ответу-фрагменту без галлюцинации, и её остается лишь указать фрагмент-галлюцинацию в ответе, ей не требуется суммаризировать данные из knowledge и исполь-

зовать собственные знания. 200 объектов было провалидировано вручную, и проверено, что автоматическое переформатирование было выполнено корректно.

4.2 Простейшие решения

В качестве базовых предсказаний рассмотрены два тривиальных подхода к классификации токенов: полное отрицание наличия галлюцинаций (Mark none, все токены помечаются как корректные) и полное их наличие (Mark all, все токены помечаются как галлюцинации). Эти методы не требуют обучения и служат нижней границей качества. Результаты представлены в таблице 2.

Метод	**HaluEval (val)**	**SemEval-500**	**SemEval-50 (Eng)**
Mark none	0.01 / 0.01	0.042 / 0.042	0.032 / 0.032
Mark all	0.289 / 0.275	0.381 / 0.351	0.376 / 0.349

Таблица 2: Macro F1 / IoU для тривиальных предсказаний на различных выборках

4.3 Рекуррентная модель

Для проведения эксперимента была выбрана реализация модели LSTM из библиотеки torch: `torch.nn.LSTM`. Размерность скрытого пространства была выбрана `hidden_size = 2048`. На выходы LSTM добавляется линейный слой `torch.nn.Linear`, понижающий размерность до 3. Обучение модели заключается в минимизации функции потерь, описанной формулой 1. В качестве данных взята валидационная выборка на всех языках от конкурса SemEval-2025. Выборка составила 500 объектов. В качестве валидационной была взята подвыборка на английском языке. Обучение состоит из 50 эпох. Использовалась платформа Kaggle.

Кривые процесса обучения представлены на графиках 4, 15. Метрики качества представлены в таблице 3. Значения метрик низкие, что объяснимо малым размером модели и тем, что подход не учитывает исходный запрос пользователя и семантику токенов

Модель	**SemEval-500**	**SemEval-50 (Eng)**
LSTM	0.23 / 0.155	0.25 / 0.19

Таблица 3: F1 Macro и IoU для обученной LSTM.

4.4 Instruction-based подход

В рамках эксперимента была выбрана модель Meta-Llama-3.1-8B-Instruct, представленная в 2024 году. Эта модель обучалась на данных до декабря 2023 года и обладает актуальными знаниями, что делает её одной из самых современных среди больших

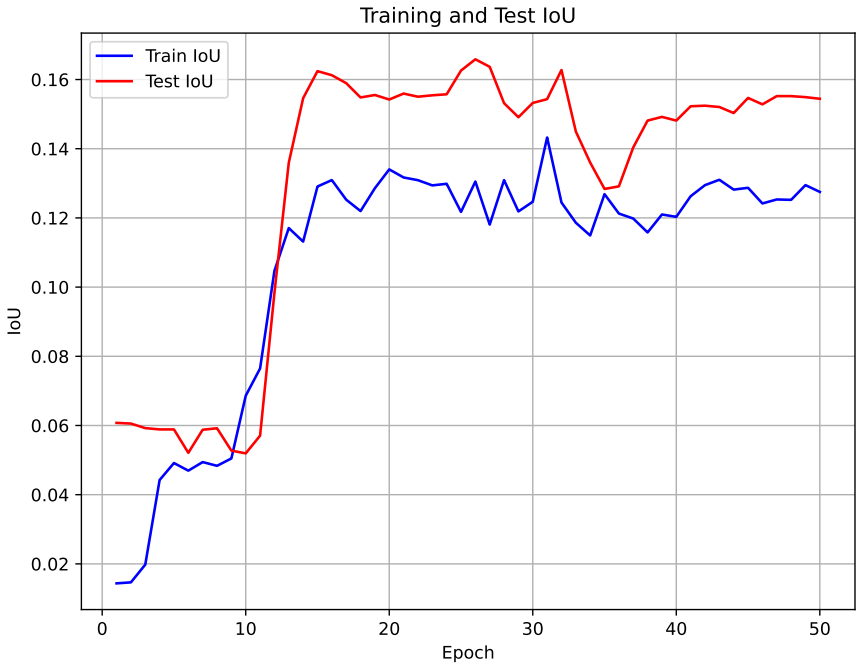

Рис. 4: График зависимости IoU от эпохи на обучающей и тестовой выборках.

языковых моделей. Модель также демонстрирует высокие результаты на стандартных бенчмарках. Важным фактором при выборе модели стало ограничение платформы Kaggle, на которой использовалась видеокарта Nvidia P100 с 16 ГБ видеопамяти, что наложило ограничения на размер модели. В связи с этим, Meta-Llama-3.1-8B-Instruct была выбрана как наиболее мощная модель, доступная для использования на данной платформе.

Для подбора оптимального промпта была взята обучающая выборка размеченного датасета HaluEval. Были проведены многочисленные эксперименты с различными промптами, и наилучшие результаты были получены при использовании промпта, приведённого в A.0.1, в этом же разделе описан и базовый промпт.

Результаты применения модели Meta-Llama-3.1-8B-Instruct представлены в таблице 4.

Модель	**HaluEval (val)**	**SemEval-500**	**SemEval-50 (Eng)**
Базовый промпт	0.356 / 0.215	0.289 / 0.178	0.298 / 0.181
Улучшенный промпт	0.745 / 0.547	0.632 / 0.331	0.675 / 0.397

Таблица 4: Macro F1 / IoU для модели Meta-Llama-3.1-8B-Instruct с различными промптами на различных выборках

При использовании базового промпта метрики качества были существенно ниже, что объясняется отсутствием четких инструкций для модели, что ограничивало её способность эффективно выявлять фактические ошибки. С новым промптом модель была снабжена дополнительными указаниями по точному определению галлюцинаций,

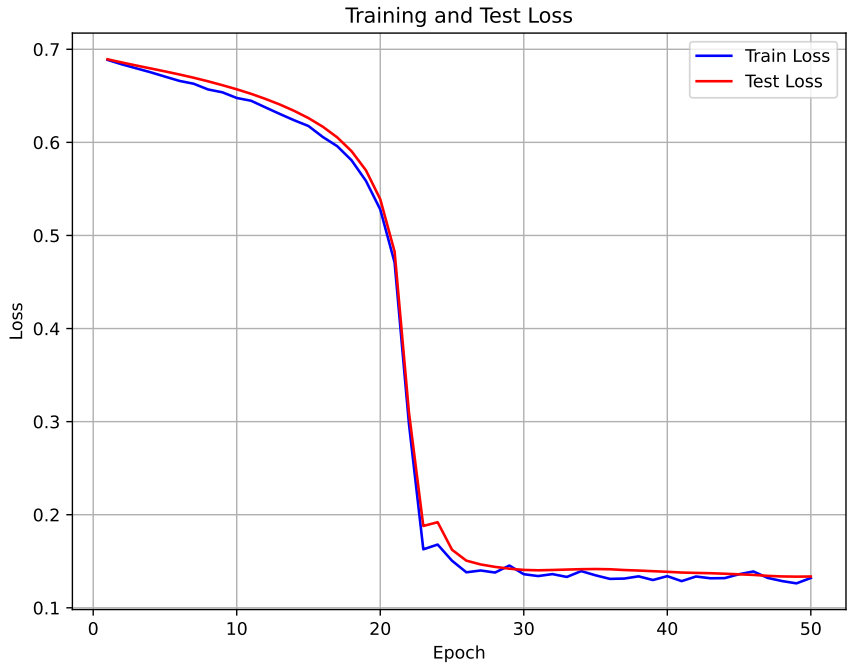

Рис. 5: График зависимости BCE Loss от эпохи на обучающей и тестовой выборках.

акцентируя внимание на фактах, датах, числах и местах, а также требуя выделять только слова, являющиеся галлюцинациями, без соседних слов. Это позволило модели более точно выполнять задачу.

Дополнительно были проведены эксперименты с моделью Qwen2.5-72B, представленной командой Alibaba в 2024 году. Эта модель имеет значительно больший объём параметров. Модель была развернута на сервере факультета ВМК МГУ. Благодаря своему масштабу и способности учитывать сложные контексты, Qwen2.5-72B превзошла Meta-Llama-3.1-8B-Instruct на всех рассмотренных подвыборках, особенно при использовании улучшенного промпта.

Модель	**HaluEval (val)**	**SemEval-500**	**SemEval-50 (Eng)**
Llama-3.1 (базовый)	0.356 / 0.215	0.289 / 0.178	0.298 / 0.181
Qwen2.5 (базовый)	0.391 / 0.283	0.330 / 0.229	0.331 / 0.234
Llama-3.1 (улучш.)	0.745 / 0.547	0.632 / 0.331	0.675 / 0.397
Qwen2.5 (улучш.)	**0.764 / 0.566**	**0.655 / 0.352**	**0.689 / 0.412**

Таблица 5: Macro F1 / IoU различных моделей с улучшенным промптом на различных выборках

4.5 Дообучение NER модели

Для решения задачи детекции галлюцинаций была дообучена модель распознавания именных сущностей (NER) на основе переразмеченного датасета HaluEval. Он включал 10 000 объектов в формате классификации токенов: пары (вопрос, ответ, фрагменты

галлюцинации). Данные были разделены в соотношении 9:1 на обучающую и тестовую выборки.

Обучение проводилось для двух архитектур: `distilbert-base-uncased` и `xlm-roberta-base`. Первая является компактной англоязычной моделью, в то время как вторая — более мощной мультиязычной, что обеспечивает ей устойчивость к разнообразным языковым конструкциям. Для обеих моделей использовались одинаковые параметры батча (`per_device_train_batch_size` = 16) и весового затухания (`weight_decay` = 0.01), однако число эпох и скорость обучения отличались: `distilbert-base-uncased` обучалась в течение 3 эпох с разогревом в 500 шагов, а `xlm-roberta-base` — 5 эпох при скорости обучения $2 \cdot 10^{-5}$.

Результаты на тестовой выборке отражены в Таблицах 6 и 7. Модель `xlm-roberta-base` показала лучшие результаты по всем метрикам: Accuracy = 0.96, F1 Macro = 0.856, F1 Micro = 0.953, IoU = 0.743 (по сравнению с 0.94 / 0.810 / 0.939 / 0.711 для `distilbert-base-uncased`). Более высокие значения F1 особенно на классах B-HALL и I-HALL демонстрируют лучшую способность модели XLM-R точно локализовать галлюцинации в тексте.

Графики изменения функции потерь и F1 Macro по шагам обучения представлены на Рисунках 6 и 7. Видно, что `xlm-roberta-base` обеспечивает более стабильное обучение и достигает лучших результатов на валидации.

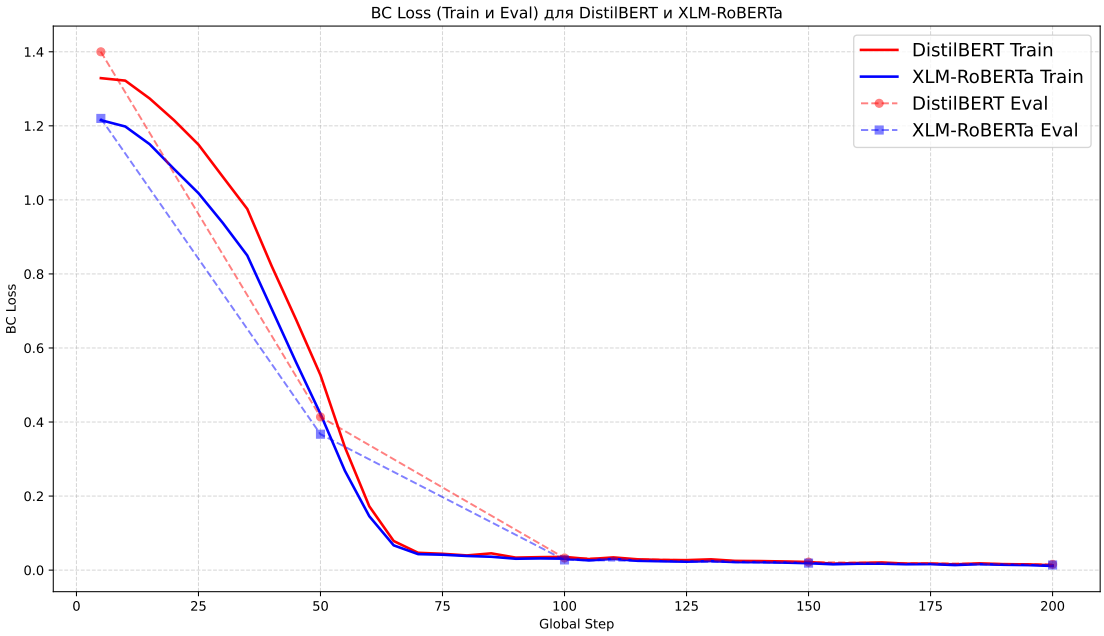

Рис. 6: Изменение значения функции потерь BCE в процессе обучения для обеих моделей.

4.6 Предложенный метод: Конвейер обработки ответов

Обнаружение галлюцинаций оценивалось с использованием автономной языковой модели Qwen2.5-72B [**?**]. В качестве основы промпта использовался улучшенный промпт, отобранный на основе экспериментов из раздела выше (см. Таблицу 5).

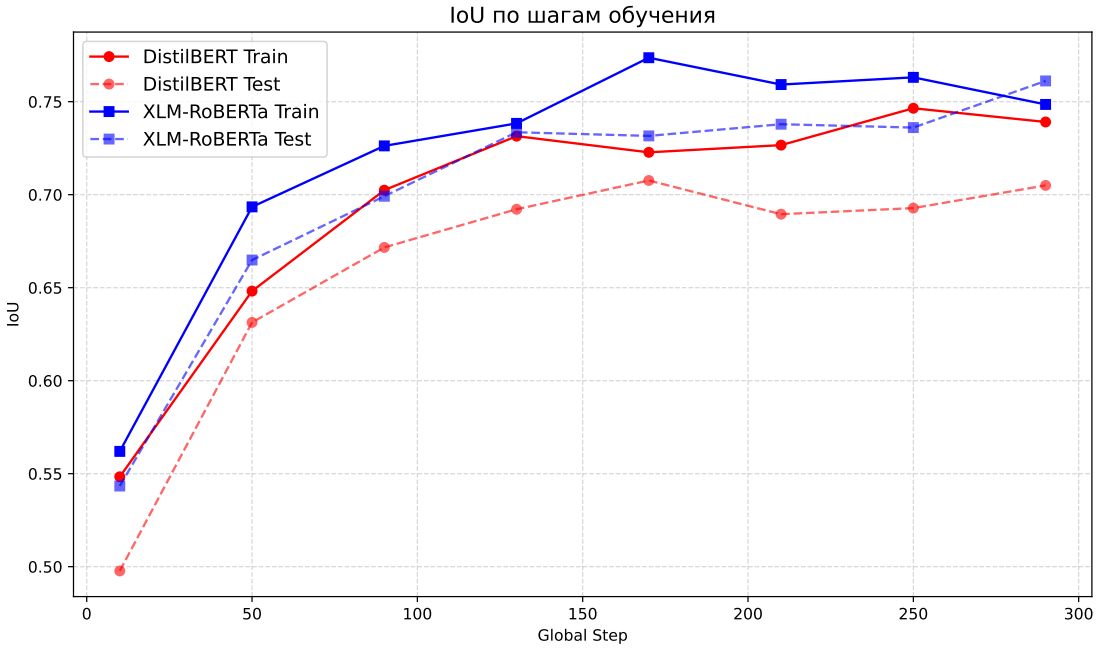

Рис. 7: Значение метрики F1 Macro на валидации по шагам обучения.

	Precision	Recall	F1-Score	Support
B-HALL	0.81	0.69	0.75	2031
I-HALL	0.78	0.62	0.69	1879
O	0.97	0.99	0.98	34945
Macro Avg	0.85	0.77	0.81	38855
Weighted Avg	0.94	0.95	0.95	38855

Таблица 6: Классификационный отчёт для BIO-разметки

Формулировки использованных подсказок приведены в Приложении A.0.1 и Приложении A.0.1.

4.6.1 Конвейер RAG

Для улучшения обнаружения галлюцинаций реализован конвейер RAG, используя Qwen2.5-72B в качестве LLM и векторную базу данных. Эксперименты проводятся на обучающей выборке HaluEval.

Корпус документов и база данных для RAG

Для повышения точности выявления галлюцинаций в предлагаемом методе используется Retrieval-Augmented Generation (RAG) с подключением внешних источников знаний. В качестве основной фактологической базы выбрано англоязычное подмножество Wikipedia [**?**], содержащее 6.41 млн статей. Перед использованием данные проходят предварительную обработку: удаляются гиперссылки, повторяющиеся символы новой строки, а также иные нерелевантные элементы, потенциально ухудшающие качество

	Precision	Recall	F1-Score	Support
B-HALL	0.86	0.72	0.78	2031
I-HALL	0.83	0.66	0.74	1880
O	0.96	0.99	0.98	34944
Macro Avg	0.88	0.79	0.83	38855
Weighted Avg	0.95	0.95	0.95	38855

Таблица 7: Классификационный отчёт модели `xlm-roberta-base` на тестовой выборке с BIO-разметкой

поиска.

Для эффективного извлечения информации применяется векторная база данных Qdrant[3], обеспечивающая быстрый и масштабируемый поиск по схожести. Векторные представления текстов формируются с использованием модели Multilingual-E5-Large [**?**]. Чтобы оптимизировать процесс векторизации документов, эмбеддинг создается на основе первых 512 символов статьи Wikipedia (а не всего текста статьи).

Сходство между входным запросом и сохранёнными векторными представлениями вычисляется с использованием косинусного расстояния, а для ускорения поиска применяется алгоритм Hierarchical Navigable Small World (HNSW), обеспечивающий баланс между точностью и быстродействием. Схема работы процесса представлена на рис. 8.

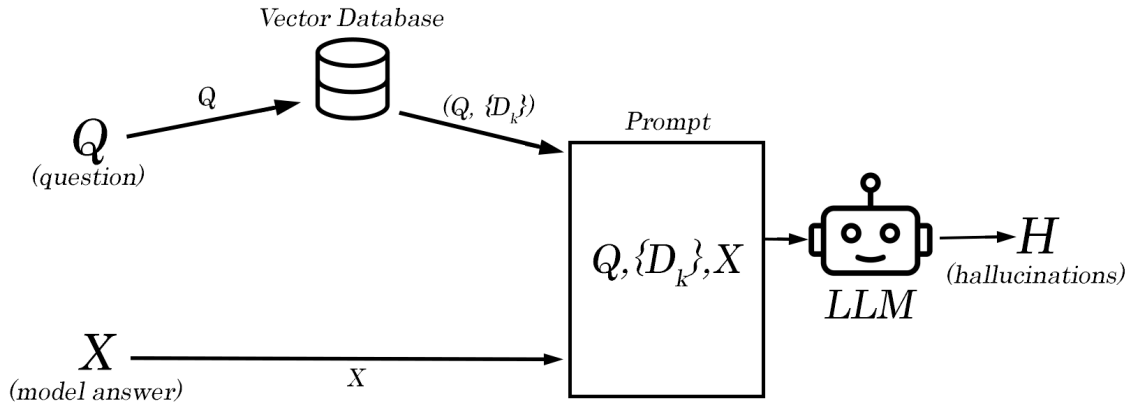

Рис. 8: Схема процесса с методом информационного поиска для детекции галлюцинаций

Эксперимент 1: Начальные промпты с Top-N документами

В начальной конфигурации использовалась подсказка, приведённая в Приложении A.0.1, с включением только одного наиболее релевантного документа (Top-1) из базы знаний. Эта установка обеспечила значение метрики IoU на уровне 52%.

Для уточнения оптимального количества поддерживающих документов была проведена серия экспериментов на обучающей выборке HaluEval, в которых варьировалось число извлекаемых документов от 1 до 10. Результаты показали, что наилучшее качество

[3] https://qdrant.tech/

достигается при использовании двух документов. Дальнейшее увеличение их количества приводило к снижению точности, вероятно, из-за увеличения объёма нерелевантной информации в контексте.

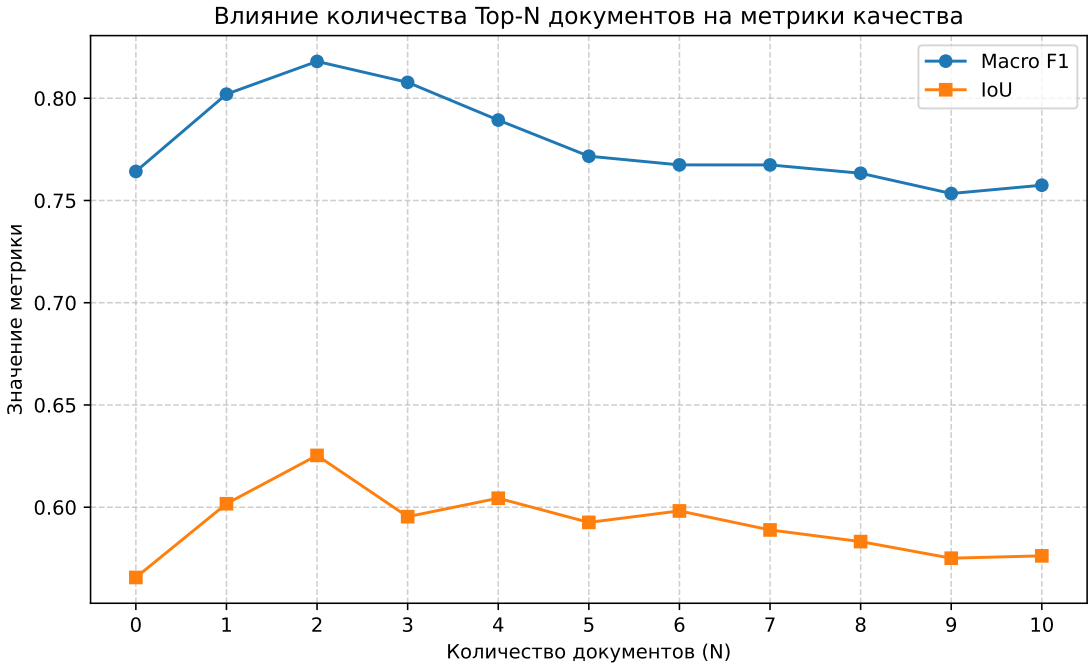

Рис. 9: Зависимость качества конвейера от кол-ва релевантных документов

Эксперимент 2: Случайный выбор Few-shot примеров

В данном эксперименте в подсказку добавляются few-shot примеры, случайно выбранные из обучающей части датасета HaluEval. В каждом примере указаны как входная реплика, так и список галлюцинаций. Цель эксперимента — определить оптимальное число few-shot примеров и количество retrieved-документов, сопровождающих каждый пример.

Варьируются два параметра: количество few-shot примеров (от 1 до 5) и количество retrieved-документов на каждый пример (от 1 до 3). Для оценки используется тренировочная часть HaluEval, где рассчитываются значения метрик Macro F1 и IoU. Наилучшие результаты достигаются при использовании трёх примеров, каждый из которых сопровождается одним документом.

Эксперимент 3: Similarity-Based Retrieval для Few-Shot

В отличие от случайного выбора few-shot примеров, в данном эксперименте используется семантический отбор: примеры выбираются из обучающей выборки HaluEval на основе сходства с текущей входной репликой. Для этого используется косинусная близость между эмбеддингами. Такой подход позволяет выбирать более релевантные примеры, которые имеют структурные и содержательные сходства с текущим запро-

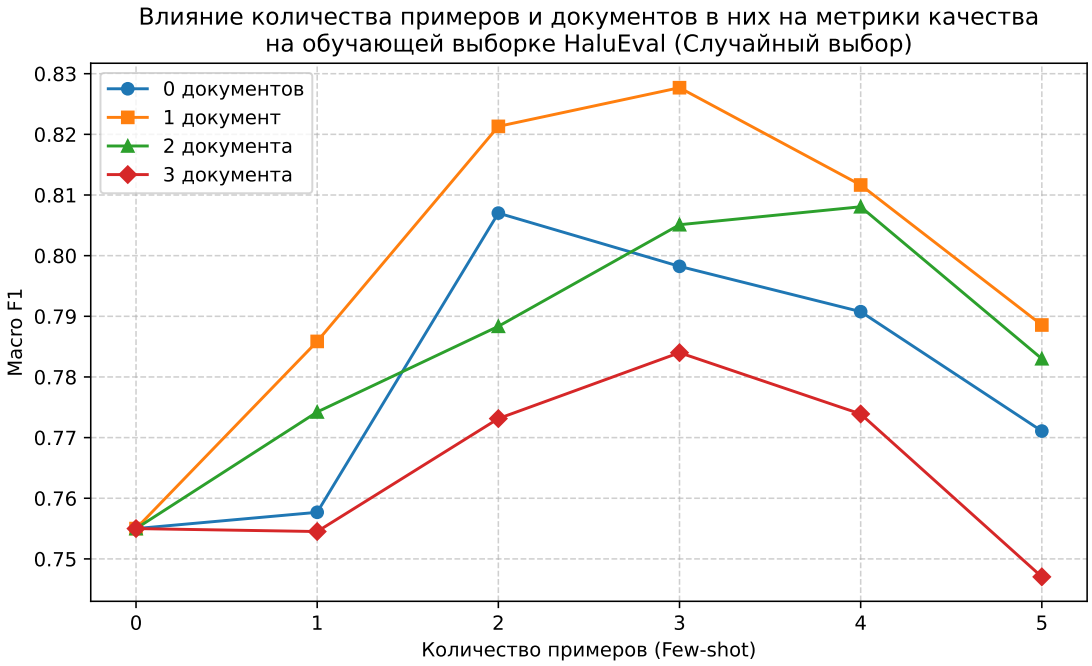

Рис. 10: Случайный выбор. Зависимость качества от кол-ва примеров и документов

сом, что потенциально улучшает способность модели к обобщению. Схема процесса с подбором примеров представлена на рис. 11.

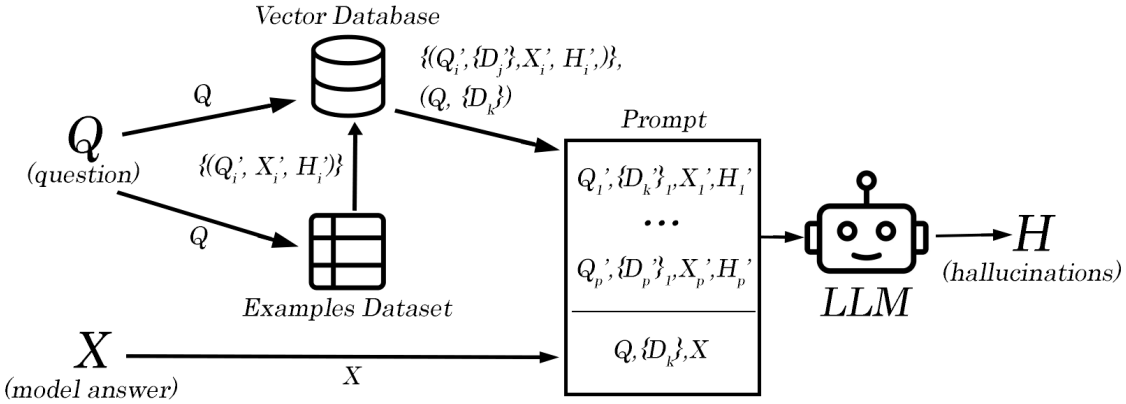

Рис. 11: Схема процесса, расширенного подбором примеров

Как и в предыдущем эксперименте, перебираются количество few-shot примеров (от 1 до 5) и количество retrieved-документов (от 1 до 3) на каждый пример. Для оценки используется обучающая часть HaluEval. Результаты эксперимента представлены на графике 12. Наилучшие результаты достигаются при использовании трёх примеров, каждый из которых сопровождается одним документом, что подтверждает эффективность similarity-based выбора.

Полученные результаты демонстрируют значительное улучшение метрик по сравнению со случайным отбором few-shot примеров (см. Таблицу **??**). Наилучшая конфигурация — три примера по одному документу на каждый — показывает Macro F1 = 0.860, что выше, чем максимум, достигнутый при случайной инициализации (Macro F1

Рис. 12: Similarity-Based. Зависимость качества от кол-ва примеров и документов

= 0.821).

Такой прирост объясняется тем, что примеры, выбранные по семантическому сходству, содержат релевантные формулировки и документы, близкие к полученному запросу пользователя, что упрощает решение задачи для модели.

Эксперимент 4: Diverse-Sampling для Few-Shot

Diverse Sampling [6]: в этом эксперименте примеры отбираются таким образом, чтобы охватывать как можно более разнообразные объекты.

В экспериментах метод показал себя хуже случайного выбора и выбора наиболее близких. Это объясняется тем, что хотя выбранные примеры более разнообразны, они не всегда являются наиболее релевантными для конкретной задачи, что снижает точность модели.

Лучшие результаты в данном эксперименте были достигнуты при использовании нуля примеров, что говорит о том, что хотя и разнообразные, но нерелевантные примеры лишь сбивают модель. На графике 13 приведены полученные метрики для различных конфигураций примеров.

4.7 Итоговое решение

В итоговом решении по результатам всех проведённых экспериментов было выбрано сочетание модели **Qwen2.5-72B**, подхода **RAG** (Retrieval-Augmented Generation) и метода **Similarity Sampling** для **few-shot learning**.

Основываясь на анализе различных вариантов выборки документов, было установле-

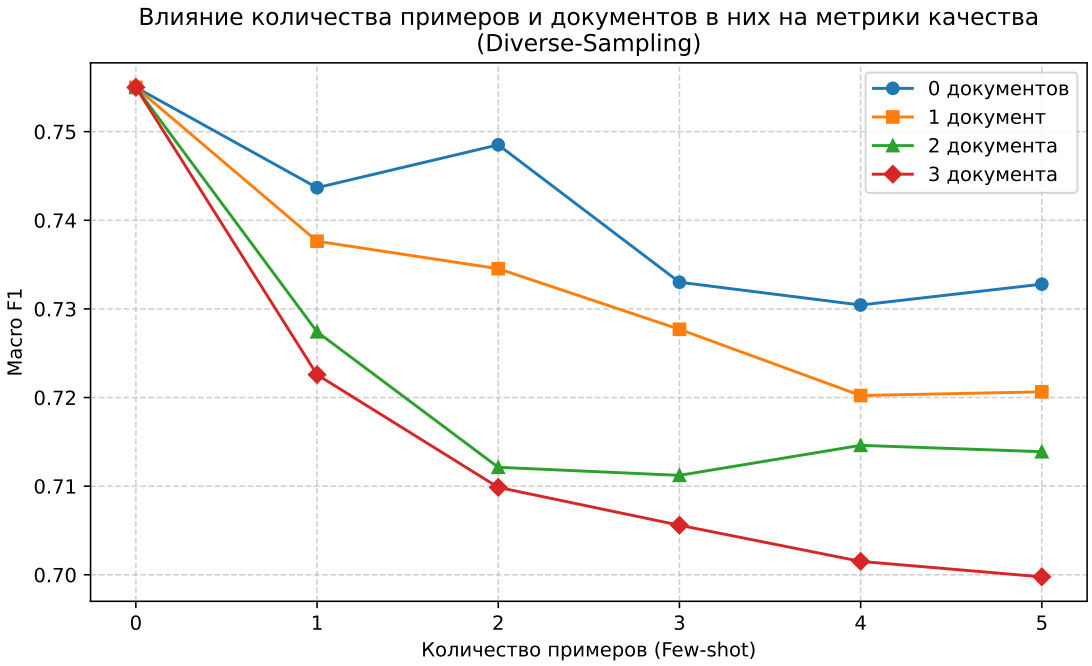

Рис. 13: Similarity-Based. Зависимость качества от кол-ва примеров и документов

но, что наиболее эффективным является использование двух документов для самого запроса и по одному документу для каждого примера. Такой подход позволил добиться наилучших показателей метрик **IoU** и **Macro F1**, что свидетельствует о высоком качестве выделения релевантных документов и точности модели в контексте выявления галлюцинаций.

Таким образом, комбинированный подход **Qwen2.5 + RAG + Similarity Sampling (few-shot)** с оптимизированной выборкой документов продемонстрировал улучшения по сравнению с предыдущими подходами. Использование двух документов для запроса позволяет модели лучше понимать контекст и извлекать более точную информацию, в то время как ограничение на один документ для каждого примера помогает избежать излишней информации и сосредоточиться на ключевых фактах.

Этот подход можно считать оптимальным решением для задач, требующих высокой точности в детекции галлюцинаций, и его можно рекомендовать для дальнейших экспериментов в подобных областях.

4.8 Сравнение рассмотренных методов

В таблице 8 представлены результаты работы моделей LSTM, Llama-3.1-8B, DistilBERT, а также предложенного подхода на различных датасетах. Для оценки использованы метрики F1 Macro и IoU.

Анализ таблицы 8 позволяет сделать следующие выводы:

- Модель Qwen2.5 в сочетании с механизмом извлечения внешней информации (RAG) и стратегией similarity-based sampling продемонстрировала стабильную и высокую

Модель	HaluEval (val)	SemEval-500	SemEval-50 (Eng)
LSTM	–	0.23 / 0.155	0.25 / 0.19
XLM-RoBERTa	0.856 / 0.743	0.327 / 0.295	0.377 / 0.344
Qwen2.5-72B	0.764 / 0.566	0.655 / 0.352	0.689 / 0.412
Qwen2.5 + RAG + Sim. sampling	**0.872 / 0.752**	**0.713 / 0.501**	**0.796 / 0.559**

Таблица 8: F1 Macro и IoU для всех моделей на различных датасетах

производительность на всех рассматриваемых датасетах, включая SemEval-500 и SemEval-50. Это свидетельствует о её высокой устойчивости к разнообразию входных данных и способности адаптироваться к различным задачам, что особенно важно при решении прикладных задач с использованием неструктурированных текстов.

- Модель XLM-RoBERTa показала лучшие результаты на валидационной части набора данных HaluEval, которая имеет схожее распределение с обучающей выборкой. Это подтверждает её хорошую способность к обобщению на подобные данные. Однако её эффективность существенно снижается при переходе к новым, более разнообразным датасетам, таким как SemEval-500 и SemEval-50, что указывает на ограниченную универсальность модели. В отличие от неё, предложенный подход на основе модели Qwen2.5 демонстрирует высокую производительность как на близких, так и на ранее не встречавшихся данных, что делает его более надёжным и применимым в широком спектре задач.

- Модель на основе LSTM существенно уступает современным трансформерным архитектурам как по метрике F1 Macro, так и по показателю IoU. Её использование не представляется целесообразным в задачах детекции галлюцинаций, где критически важна как точность, так и способность к переносу знаний на новые типы данных.

- Предложенный метод на основе Qwen2.5, усиленный компонентами RAG и отбором примеров по семантической близости, продемонстрировал наилучшие результаты среди всех рассмотренных подходов. Это объясняется способностью модели учитывать контекст и эффективно извлекать релевантную информацию из внешних источников, что позволяет существенно повысить точность локализации галлюцинаций. Высокие результаты на различных типах датасетов подтверждают практическую применимость подхода в условиях реального использования.

5 Заключение

5.1 Результаты, выносимые на защиту

- Предложен новый процесс обработки данных, обеспечивающий высокое качество детекции фактологических галлюцинаций в задачах без источника, отличающийся совместным использованием больших языковых моделей и векторных баз данных, объединяющий методы информационного поиска и автоматического подбора примеров.

- Продемонстрировано преимущество в качестве данного процесса по сравнению с известными методами (рекуррентные модели, NER модели) и подходами, использующими лишь один из блоков процесса.

5.2 Дальнейшие исследования

Дальнейшие исследования могут быть направлены на интеграцию предложенного пайплайна с узкоспециализированными дообученными языковыми моделями (например, для медицинских, юридических или научных задач) для повышения точности детекции галлюцинаций в специфических доменах. Важным шагом станет расширение функциональности пайплайна для детекции других типов галлюцинаций, таких как контекстные и внутренние ошибки, что позволит охватить более широкий спектр проблем.

Кроме того, необходимо исследовать применимость подхода в реальных приложениях, таких как чат-боты, для оценки его устойчивости к различным типам входных данных. Разработка методов улучшения интерпретируемости результатов, включая визуализацию и объяснение причин классификации текста как галлюцинации, повысит доверие пользователей. Также важно изучить влияние различных типов и объемов данных на качество детекции, включая эксперименты с мультиязычными данными и данными из разных доменов. Наконец, перспективным направлением является создание методов для онлайн-обучения и адаптации пайплайна в реальном времени, что позволит системе улучшать точность по мере получения новых данных.

Список литературы

[1] Agarwal, V., Jin, Y., Chandra, M., Choudhury, M.D., Kumar, S., Sastry, N.: Medhalu: Hallucinations in responses to healthcare queries by large language models. arXiv preprint arXiv:2409.19492 (2024), https://arxiv.org/abs/2409.19492

[2] AI, T.: Accurate hallucination detection with ner (2024), https://trieve.ai/accurate-hallucination-detection-with-ner

[3] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020), https://arxiv.org/abs/2005.14165

[4] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners (2020), https://arxiv.org/abs/2005.14165

[5] Cao, M., Dong, Y., Cheung, J.C.K.: Hallucinated but factual! inspecting the factuality of hallucinations in abstractive summarization. arXiv preprint arXiv:2109.09784 (2021), https://arxiv.org/abs/2109.0f9784

[6] Chen, R., Zhu, X., Cheng, H., Liang, P., Hashimoto, T.B.: Diversity-promoting in-context learning. arXiv preprint arXiv:2402.03038 (2024), https://arxiv.org/abs/2402.03038

[7] Dahl, M., Magesh, V., Suzgun, M., Ho, D.E.: Large legal fictions: Profiling legal hallucinations in large language models. Journal of Legal Analysis **16**(1), 64–93 (Jan 2024). doi:10.1093/jla/laae003, http://dx.doi.org/10.1093/jla/laae003

[8] Dziri, N., Madotto, A., Zaiane, O., Bose, A.J.: Neural path hunter: Reducing hallucination in dialogue systems via path grounding. arXiv preprint arXiv:2104.08455 (2021), https://arxiv.org/abs/2104.08455

[9] Ferdaus, M.M., Abdelguerfi, M., Ioup, E., Niles, K.N., Pathak, K., Sloan, S.: Towards trustworthy ai: A review of ethical and robust large language models (2024), https://arxiv.org/abs/2407.13934

[10] Filippova, K., Alt, C.: A token-level reference-free hallucination detection benchmark for free-form text generation. arXiv preprint arXiv:2104.08704 (2023), https://arxiv.org/abs/2104.08704

[11] Gao, T., Chen, D.: Ragscore: Evaluating factuality in retrieval-augmented generation. arXiv preprint arXiv:2305.14251 (2023), https://arxiv.org/abs/2305.14251

[12] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**, 1735–80 (12 1997). doi:10.1162/neco.1997.9.8.1735

[13] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Chen, D., Dai, W., Chan, H.S., Madotto, A., Fung, P.: Survey of hallucination in natural language generation. arXiv preprint arXiv:2202.03629 (2024), https://arxiv.org/abs/2202.03629

[14] Kang, H., Liu, X.Y.: Deficiency of large language models in finance: An empirical examination of hallucination (2023), https://arxiv.org/abs/2311.15548

[15] Li, J., Cheng, X., Zhao, W.X., Nie, J.Y., Wen, J.R.: Halueval: A large-scale hallucination evaluation benchmark for large language models. arXiv preprint arXiv:2305.11747 (2023), https://arxiv.org/abs/2305.11747

[16] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing (2021), https://arxiv.org/abs/2107.13586

[17] Liu, T., Zhang, Y., Brockett, C., Mao, Y., Sui, Z., Chen, W., Dolan, B.: A token-level reference-free hallucination detection benchmark for free-form text generation. arXiv preprint arXiv:2104.08704 (2021), https://arxiv.org/abs/2104.08704

[18] Liu, X., Yu, Y., Dohan, D., Chowdhery, A., Bosma, M., Wei, J., Zhou, D., Le, Q.V., Chen, E.H.: Lost in the middle: How language models use long contexts. arXiv preprint arXiv:2307.03172 (2023), https://arxiv.org/abs/2307.03172

[19] Min, S., Lewis, P., Hajishirzi, H., Zettlemoyer, L.: Rethinking the role of demonstrations: What makes in-context learning work? arXiv preprint arXiv:2202.12837 (2022), https://arxiv.org/abs/2202.12837

[20] Mu, Q., Wang, A., Durmus, E., et al.: A survey of hallucination in natural language generation. arXiv preprint arXiv:2307.13852 (2023), https://arxiv.org/abs/2307.13852

[21] OpenAI, Achiam, J., et al., S.A.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2024), https://arxiv.org/abs/2303.08774

[22] Pal, A., Umapathi, L.K., Sankarasubbu, M.: Med-halt: Medical domain hallucination test for large language models (2023), https://arxiv.org/abs/2307.15343

[23] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019), https://api.semanticscholar.org/CorpusID:160025533

[24] Rebuffel, C., Roberti, M., Soulier, L., Scoutheeten, G., Cancelliere, R., Gallinari, P.: Controlling hallucinations at word level in data-to-text generation. arXiv preprint arXiv:2102.02810 (2021), https://arxiv.org/abs/2102.02810

[25] Reynolds, L., McDonell, K.: Prompt programming for large language models: Beyond the few-shot paradigm (2021), https://arxiv.org/abs/2102.07350

[26] Sarmah, B., Zhu, T., Mehta, D., Pasquali, S.: Towards reducing hallucination in extracting information from financial reports using large language models (2023), https://arxiv.org/abs/2310.10760

[27] Strobelt, H., et al.: Developing a reliable, general-purpose hallucination detection and mitigation service. arXiv preprint arXiv:2407.15441 (2024), https://arxiv.org/abs/2407.15441

[28] Sun, Y., Du, X., al., o.: Factscore: Fine-grained hallucination detection for knowledge-grounded generation. arXiv preprint arXiv:2305.14235 (2023), https://arxiv.org/abs/2305.14235

[29] Sun, Y., Yin, Z., Guo, Q., Wu, J., Qiu, X., Zhao, H.: Benchmarking hallucination in large language models based on unanswerable math word problem (2024), https://arxiv.org/abs/2403.03558

[30] Wang, S., Wang, X., Mei, J., Xie, Y., Muarray, S., Li, Z., Wu, L., Chen, S.Q., Xiong, W.: Developing a reliable, general-purpose hallucination detection and mitigation service: Insights and lessons learned. arXiv preprint arXiv:2407.15441 (2024), https://arxiv.org/abs/2407.15441

[31] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E.H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., Fedus, W.: Emergent abilities of large language models (2022), https://arxiv.org/abs/2206.07682

[32] Yang, B., He, S., Li, T., Sun, X.: Prototype-guided in-context learning. arXiv preprint arXiv:2303.07502 (2023), https://arxiv.org/abs/2303.07502

[33] Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W.W., Salakhutdinov, R., Manning, C.D.: Hotpotqa: A dataset for diverse, explainable multi-hop question answering (2018), https://arxiv.org/abs/1809.09600

[34] Ye, T., Tian, Y., Huang, M.: Uncertainty-aware in-context learning. arXiv preprint arXiv:2302.11042 (2023), https://arxiv.org/abs/2302.11042

[35] Zha, Y., Yang, Y., Li, R., Hu, Z.: Alignscore: Evaluating factual consistency with a unified alignment function. arXiv preprint arXiv:2305.16739 (2023), https://arxiv.org/abs/2305.16739

[36] Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X., Zhao, E., Zhang, Y., Chen, Y., Wang, L., Luu, A.T., Bi, W., Shi, F., Shi, S.: Siren's song in the ai ocean: A survey on hallucination in large language models (2023), https://arxiv.org/abs/2309.01219

[37] Zhao, T., Wei, M., Preston, J.S., Poon, H.: Automatic calibration and error correction for large language models via pareto optimal self-supervision. arXiv preprint arXiv:2306.16564 (2024), https://arxiv.org/abs/2306.16564

A Приложение

Инструкция для разметчиков Semeval

- Carefully read the answer text.
- Highlight each span of text in the answer text that is not supported by the provided context (i.e., contains an overgeneration or hallucination).
- Your annotations should include only the minimum number of characters in the text that should be edited/deleted to provide a correct answer (in the case of Chinese, these will be "character components").
- You are encouraged to annotate conservatively and focus on content words rather than function words. This is not a strict guideline, and you should rely on your best judgments.
- If the answer text does not contain a hallucination, write "NO HALLUCINATION" in the comment box.
- If you are unsure about how to annotate an example, write "UNSURE" in the comment box. Please only use this option as a last resort.
- Ensure that you double-check your annotations. From the "See previous annotations" link, you can edit or delete previous annotations.

Промпты для формирования датасета HaluEval

Листинг 4: Промпт для переформатирования датасета HaluEval

```
{
  "role": "system",
  "content": "You are a fact-checking assistant. Your task is to
     identify fragments of the response that are hallucinations
     parts of the text that are factually incorrect or made up by
     model. Pay attention to facts, dates, numbers, places. Detect
     only hallucination words, without neighbour words. Give me only
     a list of fragments-hallucinations you found in
     hallucinated_answer."
}
{
  "role": "user",
  "content": "question: {item['question']}
    right_answer: {item['right_answer']}
    hallucinated_answer: {item['hallucinated_answer']}"
}
```

I want you act as a hallucination answer generator. Given a question, right answer, and related knowledge, your objective is to write a hallucinated answer that sounds plausible but is factually incorrect. You SHOULD write the hallucinated answer using the following method (each with some examples):

You are trying to answer a question but there is a factual contradiction between the answer and the knowledge. You can fabricate some information that does not exist in the provided knowledge.

#Knowledge#: The nine mile byway starts south of Morehead, Kentucky and can be accessed by U.S. Highway 60. Morehead is a home rule-class city located along US 60 (the historic Midland Trail) and Interstate 64 in Rowan County, Kentucky, in the United States.
#Question#: What U.S Highway gives access to Zilpo Road, and is also known as Midland Trail?
#Right Answer#: U.S. Highway 60
#Hallucinated Answer#: U.S. Highway 70

You are trying to answer a question but you misunderstand the question context and intention.

<Demonstrations>

You are trying to answer a question but the answer is too general or too specific to answer the question at an appropriate level of specificity.

<Demonstrations>

You are trying to answer a question but the answer cannot be inferred from the knowledge. You can incorrectly reason with the knowledge to arrive at a hallucinated answer.

<Demonstrations>

You should try your best to make the answer become hallucinated. #Hallucinated Answer# can only have about 5 more words than #Right Answer#.

#Knowledge#: <insert the related knowledge>
#Question#: <insert the question>
#Right Answer#: <insert the right answer to the question>
#Hallucinated Answer#:

Рис. 14: Промпт для генерации галлюцинированного ответа. Инструкция по выборке галлюцинаций для ответов на вопросы. Синим текстом обозначено описание намерения, красным - схема галлюцинации, зеленым - демонстрация галлюцинации.

Промпт для модели Llama-3.1-8B-Instruct

A.0.1 Базовый промпт

```
{
    "role": "user",
    "content": "Detect hallucinations in model_output:
    {formatted_str}"
}
```

Улучшенный промпт

Пример промпта, использованного в эксперименте:

```
{
    "role": "system",
    "content": "You are a fact-checking assistant. Your task is to
       identify fragments of the response that are hallucinations --
       parts of the text that are factually incorrect or made up by
       model. Pay attention to facts, dates, numbers, places. Detect
       only hallucination words, without neighbour words. Give me
       only a list of fragments-hallucinations you found in model
       output."
},
```

I want you act as an answer judge. Given a question, two answers, and related knowledge, your objective is to select the best and correct answer without hallucination and non-factual information. Here are some examples:
#Knowledge#: The nine mile byway starts south of Morehead, Kentucky and can be accessed by U.S. Highway 60. Morehead is a home rule-class city located along US 60 (the historic Midland Trail) and Interstate 64 in Rowan County, Kentucky, in the United States.
#Question#: What U.S Highway gives access to Zilpo Road, and is also known as Midland Trail?
#Answer 1#: U.S. Highway 60 (right answer)
#Answer 2#: U.S. Highway 70 (hallucinated answer)
#Your Choice#: The best answer is Answer 1.
...
<Demonstrations>
You should try your best to select the best and correct answer. If the two answers are the same, you can randomly choose one. If both answers are incorrect, choose the better one. You MUST select an answer from the provided two answers.

#Knowledge#: <insert the related knowledge>
#Question#: <insert the question>
#Answer 1#: <insert the hallucinated answer generated by the one-pass schema>
#Answer 2#: <insert the hallucinated answer generated by the conversational schema>
#Your Choice#:

Рис. 15: Промпт для отбора объектов GPT-судьёй.

```
{
    "role": "user",
    "content": formatted_str
}
```

Где `formatted_str` представляла собой строку следующего вида:

```
"query":{data["model_input"]}
"model_output":{data["model_output_text"]}
```

First Zero-shot prompt without RAG

You are a fact - checking assistant.Your task is to identify fragments of the response that are hallucinations–parts of the text that are factually incorrect or made up by model.Pay attention to facts, dates, numbers, places. Detect only hallucination words, without neighbour words. Give me only a list of fragments - hallucinations you found in model output. Write answer in JSON with the next structure:
{ ”hallucinations”: [”h1”, ”h2”] },
where h1 and h2 are hallucination fragments from model output.

Model output:

Second Zero-shot prompt without RAG

You are assistant for analysing model hallucinations. Your task is to extract fragments from model output, containing factually incorrect answers. You need to extract factually incorrect or inconsistent with input fragments from model output. Write answer in JSON with the next structure:

{ ”hallucinations”: [”h1”, ”h2”] },
where h1 and h2 are hallucination fragments from model output. Write in answer only JSON structure without other comments.

Model output:

Zero-shot prompt with RAG

You are a fact-checking assistant for analysing model hallucinations. Your task is to identify fragments in model output that are hallucinations - parts of the text that are factually incorrect or made up by model or inconsistent with model input. You get a user query in model input and hallucinated answer in model output. You also get a reliable relevant document from Wikipedia, pay attention to it while checking facts in hallucinated model output. Detect only hallucination words, without neighbour common, linking words. Write answer in JSON with the next structure:
{ 'hallucinations': ['h1', 'h2']},
where h1 and h2 are hallucinations from model output. Write your answer exactly in JSON structure without other symbols.
Relevant document: {doc 1}
Model input: {model input}
model output: {model output text}
Your answer:

One-shot prompt with RAG

You are a fact-checking assistant for analysing model hallucinations. Your task is to identify fragments in model output that are hallucinations - parts of the text that are factually incorrect or made up by model or inconsistent with model input. You get a user query in model input and hallucinated answer in model output. You also get a reliable relevant document from Wikipedia, pay attention to this document while checking facts in hallucinated model output. Detect only hallucination fragments, without neighbour common, linking words. Write answer in JSON with the next structure:
{'hallucinations': ['h1', 'h2']},
where h1 and h2 are hallucination fragments from model output. Write in answer only JSON structure without other comments. Here is an example of correct dialogue:
Relevant document example:
Model input example: {model input}
model output example: {model output text}
Your answer example:
{'hallucinations': [...]}

Input:
Relevant document: {doc 1}
Model input: {model input}
model output: {model output text}
Your answer:

Self-refine prompt

You are an assistant to check the correctness of detected hallucinations - hallucinations that were detected in model output, model output was generated by model input (question, given by user). Hallucinations are parts of the model input that are factually incorrect or made up by model or inconsistent with model input. detected hallucinations were detected by other model by given model input, model output and reliable relevant document from Wikipedia. You get the model input, model output, relevant document (pay attention to it while fact checking) and detected hallucinations (a Python list of strings that are hallucinations from model output). Your task is to fix errors in detected hallucinations, improve it by adding all missed hallucinations and removing all detections that are not hallucinations. Detect only hallucination fragments, without neighbour common, linking words. Write the answer in the same JSON format. Write in answer only JSON structure without any other comments. Here is an example of correct dialogue:
Relevant document №1 example :

Model input example: {model input}
model output example: {model output text}
Detected hallucinations: {detected hallucinations }
Your answer example:
{'hallucinations': [...]}

Input:
Relevant document №1: {doc 1}
...
Relevant document №5: {doc 5}
Model input: {model input}
model output: {model output text}
Detected hallucinations: {detected hallucinations }
Your answer: