Министерство науки и высшего образования Российской Федерации
Московский физико-технический институт
(национальный исследовательский университет)
Физтех-школа прикладной математики и информатики
Магистерская программа
«Методы и технологии искусственного интеллекта»

Аушев Ислам Амарханович

# Разработка метода гибридного информационного поиска для RAG систем

Обзор исследований

**Научный руководитель:**
Коновалов Василий Павлович

Москва

2024 г.

# Содержание

**1 Введение** **3**

**2 Постановка задачи и цель работы** **4**

**3 Обзор** **5**

3.1 Разреженные модули поиска . . . . . 6

3.1.1 TF-IDF . . . . . 6

3.1.2 BM25 . . . . . 8

3.1.3 SPLADE . . . . . 9

3.2 Плотные модули поиска . . . . . 11

3.2.1 NV-Embed . . . . . 11

3.2.2 BGE-en-icl . . . . . 13

3.3 Гибридный информационный поиск . . . . . 14

3.3.1 Последовательная стратегия . . . . . 14

3.3.2 Параллельная стратегия . . . . . 15

3.3.3 Контекстуализация . . . . . 16

**4 Результаты** **17**

# 1 Введение

Развитие больших языковых моделей (LLM) ознаменовало собой расцвет в области обработки естественного языка (NLP), открыв беспрецедентные возможности для решения сложных задач понимания и генерации текста. Их способность генерировать связный и контекстуально релевантный текст открывает широкие перспективы в различных областях, от машинного перевода и создания контента до разработки чат-ботов и виртуальных ассистентов. Актуальность LLM обусловлена их потенциалом автоматизировать сложные задачи, требующие глубокого понимания человеческого языка, обеспечивая более естественное и эффективное взаимодействие человека с машиной. Однако, несмотря на впечатляющие результаты, LLM сталкиваются с ограничениями, связанными с их склонностью к «галлюцинациям» – генерации правдоподобного, но фактически неверного текста, а также с трудностью работы с специфическими областями знаний и актуальной информацией.

Именно поэтому парадигма Retrieval Augmented Generation (RAG) приобретает особую значимость. RAG сильно влияет на качество работы LLM, предоставляя им доступ к релевантной информации из внешних источников знаний. Интегрируя мощные механизмы поиска и извлечения информации с гибкостью генеративных моделей, RAG позволяет создавать более надежные и информативные системы.

Архитектура RAG базируется на двух фундаментальных компонентах: информационный поиск (Information Retrieval) и доверительная генерация (Trust Generation).

Информационный поиск отвечает за выбор релевантной информации из внешнего источника знаний, такого как база данных, коллекция документов или веб-ресурс. На вход модель получает запрос, представленный в виде текста или векторного представления, и возвращает набор фрагментов текста, наиболее соответствующих заданному запросу, формируя контекст запроса. Модели для информационного поиска называют поискавиками или модулями поиска.

Модель для генерации, зачастую представленная большой языковой моделью, использует извлеченную информацию, полученную для конкретного запроса, и генерирует окончательный ответ. Получая на вход исходный запрос и релевантные фрагменты текста, LLM синтезирует связный и информативный ответ, основываясь на своем внутреннем знании и контексте.

Из описанной схемы работы RAG становится ясно, что результат генерации все-

цело зависит от результатов информационного поиска, ведь если для конкретного запроса поисковик вернет нерелевантный контекст, то почти гарантированно LLM сгенерирует неправильный ответ. В связи с этим получение максимально правильного контекста для запросов является одной из ключевых задач RAG.

Модели для информационного поиска можно глобально разделить на два класса: плотные (Dense Retrievers) и разреженные (Sparse Retrievers). В данный момент модели из плотного класса демонстрируют лучшие результаты по задаче информационного поиска. Однако, эти модели достаточно медленно формируют контексты для запросов, в силу своей сложности.

Для многих прикладных задач где используется RAG необходимо отвечать на запросы не только правильно, но и максимально быстро. Для подобных задач могут быть использованы разреженные поисковики, которые представляют собой легковесные варианты информационного поиска.

На практике часто приходится использовать больше одной модели для поиска нужной информации, ведь одна модель не всегда может достаточно хорошо извлекать контекст для запроса из огромной базы данных. Подход к информационному поиску, использующий композицию нескольких моделей называется гибридный информационный поиск (Hybrid Retrieval)

# 2 Постановка задачи и цель работы

Опишем формальную постановку задачи RAG. Пусть $Q$ – множество запросов (например, вопросы от пользователей). Каждый запрос $q \in Q$ представляет собой конечную последовательность слов (токенов)

$$q = \{w_1, w_2, ..., w_n\}$$

.

Имеется база знаний $D$, из которой мы хотим извлекать релевантную для запросов информацию, элементы $D$ будем далее называть документами или контекстами. Каждый элемент $d \in D$ также как и запросы является последовательностью слов.

$R(q, D)$ – модель информационного поиска релевантных документов для запроса $q \in Q$ и базы знаний $D$. На практике $R$ чаще всего является большой языковой моделью, однако использование других методов также возможно. Функция $R$ возвращает упорядоченный по релевантности набор документов $z = \{d_1, d_2, .., d_k\}$, этот

набор документов формирует контекст для дальнейшей доверительной генерации. Имеется функция $M_1(Q, D, R)$ – метрика, оценивающая качество информационного поиска.

Имея для запроса $q \in Q$ релевантный контекст $z$, модель для генерации $G(q, z)$ возвращает ответ для запроса. Качество генерации также оценивается метрикой $M_2(Q, D, Q_{truth}, R)$, где $Q_{truth}$ – множество правильных ответов для запросов.

Задача RAG сводится к следующей оптимизационной задаче:

$$\arg\min_{R,G} M_2(Q, D, Q_{truth}, R)$$

В такой постановке задача RAG не решает оптимизационную задачу для метрики $M_1$, однако на практике чаще всего сначала находят оптимальную функцию $R$ в смысле метрики $M_1$, а затем уже функцию $G$.

Как уже было сказано ранее, качество RAG систем всецело зависит от качества информационного поиска, то есть от функции $R$. Сама функция $R$ в задаче может быть произвольной, это может быть один поисковик или композиция нескольких, связанных между собой. Однако, на практике гибридная композиция моделей показывает результаты лучше, чем использование одной модели, поэтому важно развивать методы для построения композиций.

Выше описана формальная общая постановка задачи RAG. Целью же данной работы является улучшение и развитие методов гибридного информационного поиска, то есть части задачи RAG, связанной с поиском оптимальной модели R при заданном множестве запросов $Q$ и имеющейся базе данных $D$. При этом хочется разработать методы, которые бы демонстрировали хорошее качество на ряде задач, а не для конкретных фиксированных $Q$ и $D$.

# 3 Обзор

В архитектуре RAG критическим компонентом является механизм поиска, отвечающий за извлечение релевантной информации из внешнего хранилища знаний. Существуют два основных подхода к поиску: разреженные и плотные поисковики. В свою очередь гибридный информационный поиск подразумевает собой возможность совместного использования разных модулей поиска, поэтому важно рассмотреть текущие существующие методы этих двух классов. В данном разделе мы рассмотрим

оба подхода, их преимущества и недостатки, а также рассмотрим существующие методы для построения гибридных композиций моделей.

## 3.1 Разреженные модули поиска

Разреженные модели основаны на представлении текста в виде "мешка слов"(bag-of-words), где каждый документ и запрос представляются векторами размерности длины словаря, элементы которых демонстрируют важность соответствующих слов. Ключевой особенностью является разреженность этих векторов: большинство элементов равны нулю, так как большинство слов из общего словаря отсутствуют в конкретном документе. Это позволяет эффективно хранить и обрабатывать данные.

### 3.1.1 TF-IDF

Пусть $V = \{v_1, v_2, \dots v_N\}$ упорядоченный словарь всех слов, присутствующих в базе знаний.

TF-IDF [1] (Term Frequency-Inverse Document Frequency) – это статистическая мера, используемая для оценки важности слова в документе относительно коллекции документов. Она основана на двух компонентах: Term Frequency (TF) и Inverse Document Frequency (IDF).

TF характеризует частоту слова в документе, чем чаще слово встречается в документе, тем выше его TF. Рассмотрим $v_i \in V$ и документа $d_j \in D$, где $D$ наша база данных откуда мы хотим получать контексты запросов. Тогда формула для вычисления TF выглядит следующим образом:

$$\mathrm{TF}(v_i, d_j) = \frac{n_i}{\sum_{k=1}^{N} n_k}$$

где $n_i$ – есть число вхождений слова $v_i$ в документ $d_j$, а в знаменателе — общее число слов в данном документе.

IDF измеряет насколько редко слово встречается во всей коллекции документов. Чем реже термин встречается, тем выше его IDF. Формула для IDF:

$$\mathrm{IDF}(v_i, D) = \ln \frac{N}{|\{d_j \in D | v_i \in d_j)\}|}$$

где $N$ – число документов в $D$, $|\{d_j \in D | v_i \in d_j)\}|$ – число документов в которых встречается слово $v_i$

TF-IDF определяется следующим образом:

$$\text{TF-IDF}(v_i, d_j, D) = \text{TF}(v_i, d_j) \cdot \text{IDF}(v_i, D)$$

Опишем алгоритм построения векторов для запросов и документов. Пусть есть запрос $q = \{q_1, q_2, \dots q_s\}$, тогда для каждого слова $q_i$ мы считаем TF-IDF$(q_i, q, D)$ и результат записываем в ячейку нашего вектора с номером $m$, соответствующим номеру слова $q_i$ в упорядоченном словаре $D$. Результирующий вектор будет размерности размера словаря, при чем этот вектор будет сильно разреженным, то есть там будет много нулей, потому что большинство слов из словаря не будут присутствовать в запросе. Векторные представления для документов получаются аналогично.

Имея векторные представления для всех документов и запроса, мы можем найти для запроса наиболее релевантные документы, вычисляя меру релевантности с помощью различных метрик расстояний между векторами, например косинусной меры или скалярного произведения.

У данного подхода можно выделить следующие преимущества и недостатки:

Преимущества:

1. Скорость: TF-TDF достаточно эффективно работает с очень большими базами документов.

2. Интерпретируемость: Легко понять, какие термины повлияли на релевантность документа, достаточно лишь посмотреть на соответствующие векторы

Недостатки:

1. Семантическая ограниченность: TF-IDF может плохо улавливать семантическую близость между запросом и документом, потому что опирается лишь на факт вхождения слова в запрос и документ. Если же запрос и документ используют разные слова для выражения одной и той же идеи, то TF-IDF не сможет это учесть и скорее всего допустит ошибку.

2. Лексическая ограниченность: TF-IDF не учитывает слова, которые не встречаются в исходном словаре, что усложняет поиск.

### 3.1.2 BM25

BM25 [2] (Best Match 25) -– это усовершенствованный вариант TF-IDF, который учитывает длину документа и использует настраиваемые параметры для оптимизации поиска. В отличии от TF-IDF, алгоритм BM25 по-разному считает векторные

представления для запроса и документов. Для запроса BM25 формирует вектор размерности размера словаря, где в соответсвующих словам запроса ячейках находятся значения $\text{IDF}(q_i, D)$, при этом формула для подсчета IDF немного отличается.

$$\text{IDF}(q_i, D) = \ln\left(\frac{N - |\{d_j \in D | v_i \in d_j)\}| + 0.5}{|\{d_j \in D | v_i \in d_j)\}| + 0.5} + 1\right)$$

Для документа $d$ элемент вектора с номером $i$ вычисляется по формуле:

$$\frac{f(q_i, d) \cdot (k + 1)}{f(q_i, d) + k \cdot (1 - b + b \cdot \frac{N}{A})}$$

где $f(v_i, d)$ – число вхождений слова $v_i$ в документ $d$, A – средняя длина документов в базе знаний, $k$ и $b$ – настраеваемые параметры, но зачастую полагают $k = 2, b = 0.75$ю

Используя векторные представления для запроса и документа, релевантность между ними получается в результате скалярного произведения соответствующих векторов. В результате получается следующая формула для BM25:

$$score(q, d) = \sum \text{IDF}(q_i, D) \cdot \frac{f(v_i, d) \cdot (k_1 + 1)}{f(v_i, d) + k_1 \cdot (1 - b + b \cdot \frac{N}{A})}$$

Метод BM25 имеет те же преимущества и недостатки, что и TF-IDF, при этом для большинства задач BM25 показывает себя лучше, чем TF-IDF, поэтому в качестве стартового решения для конкретной задачи часто берут именно этот алгоритм.

### 3.1.3 SPLADE

Два прошлых метода основываются на частоте слова в текстах, но никак не учитывают семантическую близость между запросами и документами. Трансформеры и большие языковые модели хорошо справляются с определением семантических связей между словами в тексте, поэтому использование этих подходов сможет решить проблему семантической ограниченности при информационном поиске. Одной из лучших моделей, реализующей эту идею, является модель SPLADE [8]. SPLADE (SParse Lexical AnD Expansion Model) – подход основанный на архитектуре модели BERT [4] и задаче Masked Language Modeling (MLM). Задача MLM заключается в распознавании замаскированных токенов исходной последовательности. Для решения задачи MLM BERT принимает на вход последовательность токенов и для каждого токена возвращает некоторое распределение над словарем, далее для замаскированных токенов предсказывается слово, стоящее на этом месте. Модель SPLADE проводит аналогичные вычисления, но никаких маскирований не производит. Для

каждого токена модель получает некоторый вектор размерности размера словаря, а далее из всех этих векторов получает один, который содержит в себе всю информацию о словах в исходном тексте.

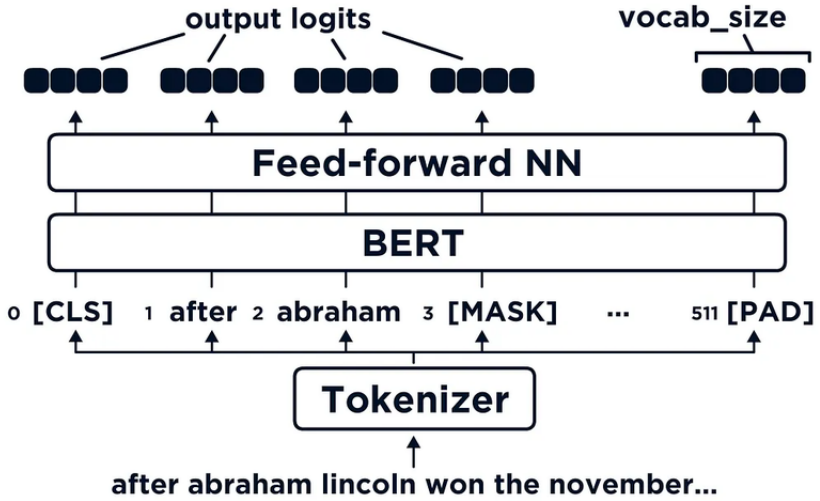


Рис. 1: Архитектура модели BERT для задачи Masked Language Modeling

Пусть векторы $h_1, \dots h_n$ – эмбеддинги полученные для каждого токена последовательности моделью BERT. Далее получаем векторы $\hat{h}_i$ с помощью линейной проекции векторов $h_i$, размерность $\hat{h}_i$ совпадает с размером словаря. Теперь с помощью последовательности $\hat{h}_1, \dots \hat{h}_n$ получим вектор $w$, который содержит в себе информацию о важности каждого слова в последовательности. Вектор $w$ считается по следующей формуле:

$$w = \sum_{i}^{n} \ln(1 + \mathrm{ReLU}(\hat{h}_i))$$

где все операции над векторами покомпонентные.

Вектор $w$ содержит в себе информацию о каждом слове из исходного запроса или документа. Используя полученные векторные представления, производится поиск наиболее релевантных документов.

Основным преимущества модели SPLADE является тот факт, что SPLADE не является семантически ограниченным, потому что семантическая связь явно закладывается в векторные представления с помощью BERT. Недостатком подхода является слабая разреженность получаемых представлений, а также сложность обучения модели, по сравнению с TF-IDF и BM25

## 3.2 Плотные модули поиска

Плотные модули поиска используют нейронные сети для создания векторных представлений документов и запросов. В отличии от разреженных моделей, эти векторные представления зачастую имеют меньшую размерность, а также не являются разреженными. Эти представления фиксируют семантическое значение текста, позволяя находить документы, близкие по смыслу к запросу, даже если они не содержат тех же самых ключевых слов. Именно по этой причине плотные модели на практике показывают более внушительные результаты, но у всех этих моделей есть свои общие недостатки.

Недостатки плотных поисковиков:

1. Интерпретируемость: Сложнее понять, почему модель считает документ релевантным.

2. Масштабируемость: Вычислительно затратные, особенно при работе с очень большими базами данных.

3. Сложность обучения: Требуют больших объемов данных для обучения.

### 3.2.1 NV-Embed

Этот подход [13] основывается на использовании LLM архитектуры для получения векторных представлений. Существует два популярных метода получения эмбеддингов для последовательности токенов: 1) mean pooling и 2) использование эмбеддинга последнего токена <EOS>. Encoder модели обычно используют mean pooling (Wang et al., 2022; Izacard et al., 2021), в то время как использование эмбеддинга токена <EOS> более популярно для LLM. Однако оба метода имеют определенные ограничения. Mean pooling просто берет среднее значение по эмбеддингам токенов и так может разбавить важную информацию из ключевых фраз текста, в то время как использование одного эмбеддинга токена <EOS> зависит всецело лишь от эмбеддинга последнего токена, что может приводить к ошибкам и неточностям. В работе авторы представили слой Latent Attention Layer для получения эмбеддинга последовательности токенов. Схема этого слоя представлена на Рис.2

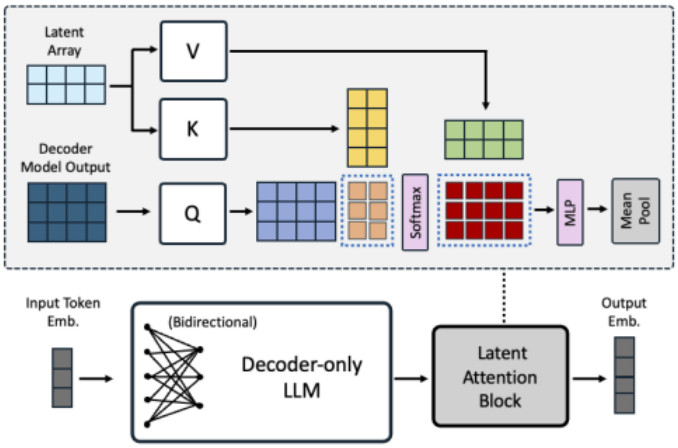


Рис.2: Latent Attention Layer.

На вход слою приходят выходы из последнего слоя декодера, эта последовательность образует матрицу, обозначим ее как $Q \in \mathbb{R}^{l \times d}$, где $l$ – длина последовательности, а $d$ – размерность эмбеддингов. У слоя есть обучаемые латентные параметры, которые представляют собой матрицу $K$ размерности $r \times d$, где $r$ – гиперпараметр. Тогда выход слоя получается следующим образом: сначала вычисляется перекресный механизм внимания между $Q$ и $K$, в результате получается матрица $O \in \mathbb{R}^{l \times d}$

$$O = softmax(Q \cdot K^\top) \cdot K$$

Затем идет два полносвязных слоя с функцией активации GeLU и усреднение эмбеддингов для получения эмбеддинга всей последовательности. В статьей авторы продемострировали, что такой подход дает прирост к качеству, по сравнению с другими стратегиями получения эмбеддинга всей последовательности.

В качестве LLM архитектуры авторы использовали Mistral 7B[10]. Обучение проходило с помощью constrastive learning. В ходе обучения авторы обнаружили, что использование causal attention маски мешает модели обучаться и предложили не использовать causal маску, тем самым переходя к обычному механизму внимания. Также авторы в работе предложили обучать модель не только для задачи информационного поиска, но и для ряда других задач, таких как кластеризация и классификация, тем самым расширяя обобщающую способность модели. Также авторы предложили улучшение модели NV-Embed-v2, которая отличается от исходной модели подбором негативных примеров для constrastive learning, с помощью другого модуля поиска. Сейчас NV-Embed-v2 занимает первое место на бенчмарке MTEB [11].

### 3.2.2 BGE-en-icl

Decoder-only LLM демонстрируют отличные возможности для in-context learning (ICL) [5, 7]. Эта функция позволяет им эффективно справляться как со знакомыми, так и с новыми задачами, используя примеры, предоставляемые в рамках промпта вместе с исходным входом. Анализируя промпт LLM начинает лучше понимать новую для нее задачу и генерировать более качественный ответ. Авторы статьи в своей модели bge-en-icl[12] предложили использовать эту идею для улучшения процесса создания эмбеддингов входных запросов и соответственно для улучшения информационного поиска. Архитектура модели bge-en-icl продемонстрирована на Рис. 3

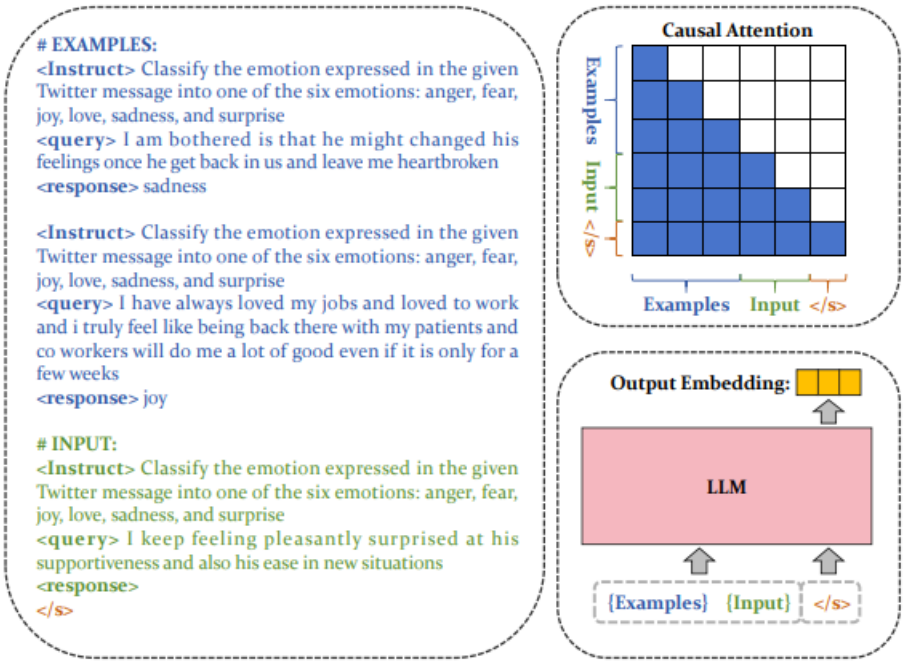


Рис 3. Архитектура bge-en-icl

Опишем схему ICL модели в рамках задачи информационного поиска. Пусть есть пара $(q_i, d_i)$ – запрос и релевантный ему документ. По этой паре формируется шаблон $\text{example}_i$, который будет использоваться далее в качестве промпта.

$$\text{example}_i = \text{<Instruct>}\{\text{task definition}\}\text{<query>}q_i\text{<response>}d_i$$

в этом шаблоне под task definition понимается текстовое описание конкретной задачи, пример такого описания для задачи классификации есть на Рис. 3

Из $n$ таких пар формируется весь промпт, он будет иметь следующий вид:

$$\text{prompt} = \{\text{example}_1\}\{\text{example}_2\}\ldots\{\text{example}_n\}\text{<Instruct>}\{\text{task definition}\}\text{<query>}q$$

где $q$ – запрос для которого мы хотим найти релевантные документы.

Для constrastive обучения используется функция InfoNCE[9]. В качестве эмбеддинга текстовой последовательности берется выход последнего слоя для токена <EOS>. В отличие от NV-Embed авторы обучают LLM с помощью стандартной causal attention маски. В статье были испробованы и другие комбинации формирования эмбеддингов последовательности и механизмов внимания, но наилучший результат на бенчмарке MTEB показала именно эта пара. В качестве декодера брался также Mistral-7B.

Несмотря на качество модели, у данного подхода есть очевидный недостаток, а именно сложность подбора самих промптов и их количества для конкретной задачи.

## 3.3 Гибридный информационный поиск

Гибридные модели представляют собой композицию нескольких моделей в одну более качественную модель, по аналогии с ансамблями из классического машинного обучения эти поисковики будем называть базовыми. Все подходы к гибридному поиску можно разделить на две стратегии: параллельные и последовательные, либо же их комбинация в рамках одной архитектуры.

### 3.3.1 Последовательная стратегия

Суть последовательной стратегии заключается в пошаговом применении базовых модулей поиска, основываясь на результате предыдущих. А именно, пусть есть запрос $q$ и мы хотим найти для него релевантный контекст из базы данных $D$. Для этого возьмем какой-нибудь алгоритм для информационного поиска и найдем набор наиболее релевантных запросу документов $\{d_1, \dots d_k\}$. Затем с помощью второго базового поисковика из этого набора найдем поднабор релевантных документов $\{\hat{d}_1, \dots \hat{d}_{k_1}\}$, где $k_1 \leqslant k$, и так далее. В результате мы получим некоторый набор документов, которые наша модель посчитает наиболее релевантными к запросу $q$.

У такого подхода есть свои плюсы и минусы:

Плюсы данного подхода:

1. Возможность более эффективного поиска документов: Базы данных бывают очень большими и использовать тяжеловесные подходы, по типу плотных модулей поиска, может оказаться очень труднозатратно. В таком случае можно использовать более легковесные подходы, например BM25, для префильтрации

всей базы данных и извлечения небольшого списка документов, откуда уже более сильные базовые модели будут извлекать релевантные контексты.

2. Более качественный поиск : Композиция моделей может лучше справиться с задачей.

Минусы:

1. Плохая отказоустойчивость : Если мы захотим поменять что-то на ранних этапах нашей архитектуры, то придется пересчитывать все результаты заново.

В основном последовательную стратегию реализовывают используя модель для информационного поиска в паре с cross-encoder или bi-encoder моделью (ColBERT[6], bge-reranker[12], и др.), которую еще принято называть моделью ранжирования. Сначала модель находит релевантные документов из базы знаний, затем cross-encoder модель переранжирует этот набор документов и вернет новый набор, тем самым перераспределяя релевантные документы по своим местам, из которых мы можем взять уже меньшее подмножество релевантных документов.

### 3.3.2 Параллельная стратегия

Параллельная стратегия подразумевает собой получение наборов релевантных документов для запроса с помощью нескольких поисковиков, а затем аггрегирование полученных результатов.

Одним из наиболее распространенных методов для объединения результатов нескольких модулей поиска в один является алгоритм Reciprocal Rank Fusion (RRF)[3]. Суть метода заключается в следующем: пусть у нас есть два набора релевантных документов $w_1, w_2$, полученных разными моделями. Тогда RRF из двух наборов делает один по следующей формуле: пусть $d \in w_1$ или $d \in w_2$, тогда

$$\text{score}(d) = \frac{1}{\text{rank d in } w_1 + K} + \frac{1}{\text{rank d in } w_2 + K}$$

где $K$ – константа, если же документа $d$ нет в $w_1$ или же в $w_2$, то соответствующего слагаемого в сумме нет. Затем результаты сортируются в порядке невозрастания и из полученного набора выбирается поднабор из наиболее релевантных. Понятно, что чем меньше был ранг у документа в наборах $w_1$ и $w_2$, тем меньше у него будет ранг в итоговом наборе. Схема RRF легко обобщается на случай $n$ базовых поисковиков.

Преимуществом RRF является простота алгоритма, недостаток алгоритма проявляется в ситуации когда какой документ отсутствует в одном из наборов $w_1$ или $w_2$, в таком случае тогда одно слагаемое будет использовано для подсчета результата, что может сказаться на финальном наборе. Однако, если одна из моделей не включила этот документов в свой набор наиболее релевантных, то это скорее всего означает, что этот документ в целом недостаточно релевантен, поэтому и второй поисковик должен изначально выставить этому документу высокий ранг.

### 3.3.3 Контекстуализация

Чтобы сформировать базы знаний для задач RAG большие объемы текстов нарезаются на части, эти части мы затем называем документами. Документы представляют собой лишь небольшую часть исходного текста и самих документов иногда бывает очень много. Затем используя различные модели для поиска релевантных документов, мы решаем задачу информационного поиска. Но здесь мы сталкиваемся с проблемой, возникшей когда формировались документы. Каждый документ есть небольшой непрерывный фрагмент из исходных текстов, существуют разные подходы к правильному разбиению исходных текстов на части, но у всех них есть недостаток. Полученные таким образом куски текста не содержат контекст который в себе подразумевают. Рассмотрим пример: нам на вход пришел запрос "Насколько сильно выросли акции компании Google в 2024 году?". Поисковик может вернуть в качестве релевантного документа следующий: "Акции компании выросли на n% в 2024 году но доверять этому ответу не стоит, потому что данный документ не имеет информации о том, о какой компании конкретно идет речь.

Чтобы исправить это используют контесктуализацию[14] документов. В момент когда мы разбиваем исходные тексты на части мы просим LLM составить краткое содержание контекста данного фрагмента текста, таким образом, полученные части уже будут иметь более полную информацию и помогут избавиться от проблемы отсутствия контекста.

Сам метод контекстуализации не представляет собой гибридную схему для информационного поиска, но этот метод приведен в этом разделе, потому что зачастую используется совмество с описанными ранее стратегиями гибридного поиска.

# 4 Результаты

Часть из рассмотренных в прошлом разделе методов были испробованы на практике в рамках Regulatory Natural Language Processing Workshop 2025, который проводится совместно с конференцией COLING 2025. В рамках задачи было необходимо создать RAG систему для датасета ObliQA. В этом наборе данных используются нормативные документы Abu Dhabi Global Markets (ADGM), которые устанавливают правила предоставления финансовых услуг в ОАЭ. База данных ObliQA состоит примерно из 640 000 слов нормативного текста в 40 документах, каждый объемом от 30 до 100 страниц. Эти документы имеют сложную внутреннюю структуру, такую как подразделы, нумерованные статьи и перекрестные ссылки.

В предложенном решении были реализованы две идеи:

1. Префильтрация исходных документов с помощью BM25, для получения более узкого набора.

2. После префильтрации каждый поисковик формирует свой набор релевантных документов, далее производится Reciprocal Rank Fusion между моделями BM25, NV-Embed-v2 и bge-en-icl.

Первая идея реализует последовательную стратегию гибридного поиска, а вторая параллельную.

Данное решение заняло 2 место на соревновании и статья с описанным подходом принята на конференцию COLING 2025.

# Список литературы


1. *Salton G.*, *McGill M.* Introduction to Modern Information Retrieval //. — 1983. — URL: https://api.semanticscholar.org/CorpusID:43685115.
2. Okapi at TREC-3 / S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, M. Gatford // Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994. Т. 500—225 / под ред. D. K. Harman. — National Institute of Standards, Technology (NIST), 1994. — С. 109—126. — (NIST Special Publication). — URL: http://trec.nist.gov/pubs/trec3/papers/city.ps.gz.
3. *Cormack G. V.*, *Clarke C. L. A.*, *Büttcher S.* Reciprocal rank fusion outperforms condorcet and individual rank learning methods // Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval. — 2009. — URL: https://api.semanticscholar.org/CorpusID:12408211.
4. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin, M.-W. Chang, K. Lee, K. Toutanova // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers) / под ред. J. Burstein, C. Doran, T. Solorio. — Minneapolis, Minnesota : Association for Computational Linguistics, 06.2019. — С. 4171—4186. — DOI: 10.18653/v1/N19-1423. — URL: https://aclanthology.org/N19-1423.
5. Language Models are Unsupervised Multitask Learners / A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever //. — 2019. — URL: https://api.semanticscholar.org/CorpusID:160025533.
6. *Khattab O.*, *Zaharia M.* ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT // Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. — Virtual Event, China : Association for Computing Machinery, 2020. — С. 39—48. — (SIGIR '20). — ISBN 9781450380164. — DOI: 10.1145/3397271.3401075. — URL: https://doi.org/10.1145/3397271.3401075.

7. Language models are few-shot learners / T. B. Brown [и др.] // Proceedings of the 34th International Conference on Neural Information Processing Systems. — Vancouver, BC, Canada : Curran Associates Inc., 2020. — (NIPS '20). — ISBN 9781713829546.

8. *Formal T.*, *Piwowarski B.*, *Clinchant S.* SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking // Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. — Virtual Event, Canada : Association for Computing Machinery, 2021. — С. 2288—2292. — (SIGIR '21). — ISBN 9781450380379. — DOI: 10.1145/3404835.3463098. — URL: https://doi.org/10.1145/3404835.3463098.

9. Unsupervised Dense Information Retrieval with Contrastive Learning / G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, E. Grave // Transactions on Machine Learning Research. — 2022. — ISSN 2835-8856. — URL: https://openreview.net/forum?id=jKN1pXi7b0.

10. Mistral 7B / A. Q. Jiang [и др.] // CoRR. — 2023. — Т. abs/2310.06825. — DOI: 10.48550/ARXIV.2310.06825. — arXiv: 2310.06825. — URL: https://doi.org/10.48550/arXiv.2310.06825.

11. MTEB: Massive Text Embedding Benchmark / N. Muennighoff, N. Tazi, L. Magne, N. Reimers // Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics / под ред. A. Vlachos, I. Augenstein. — Dubrovnik, Croatia : Association for Computational Linguistics, 05.2023. — С. 2014—2037. — DOI: 10.18653/v1/2023.eacl-main.148. — URL: https://aclanthology.org/2023.eacl-main.148.

12. *Anonymous*. Making Text Embedders Few-Shot Learners // Submitted to The Thirteenth International Conference on Learning Representations. — 2024. — URL: https://openreview.net/forum?id=wfLuiDjQ0u ; under review.

13. *Anonymous*. NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models // Submitted to The Thirteenth International Conference on Learning Representations. — 2024. — URL: https://openreview.net/forum?id=lgsyLSsDRe ; under review.

14. *Anthropic*. Introducing Contextual Retrieval. — 2024. — URL: https://www.anthropic.com/news/contextual-retrieval ; Accessed: 2024-11-29.