Temporal RAG: Адаптация Retrieval-Augmented Generation для анализа финансовых новостных потоков с учетом фактора времени

Артём Каменцев, Борис Сажин
Курс NLP 2025
Ссылка на репозиторий: `https://github.com/szhn1/itmo_nlp_project`

15 января 2026 г.

Аннотация

В данной работе представлена система **Temporal RAG**, разработанная для мониторинга и саммаризации высокочастотных новостных сообщений из Telegram (экономика/финансы). Классические RAG-пайплайны ранжируют документы в основном по семантической близости и могут «утащить» в контекст устаревшие публикации с похожей формулировкой. Мы реализовали end-to-end пайплайн, который объединяет гибридный поиск (Dense FAISS + BM25) через *Reciprocal Rank Fusion* и добавляет явный временной компонент к ранжированию относительно актуальной даты. Дополнительно применяется дедупликация/кластеризация кандидатов и LLM-фильтрация релевантности перед генерацией итогового дайджеста. Эксперименты показали рост качества итоговых саммари по оценке LLM-судьи (overall 3.90 против 3.62 у базового RAG) и заметное улучшение актуальности (*recency*).

1 Введение

Финансовые рынки крайне чувствительны к новостному фону, и ключевой характеристикой информации является её *свежесть*. Даже при корректном семантическом совпадении запросов и документов, устаревшие сообщения могут занимать значительную долю контекста и ухудшать качество итоговой сводки. Цель проекта — построить RAG-систему, которая:

- сохраняет высокую семантическую релевантность контекста;
- явно учитывает время публикации документов и опорную дату запроса (`anchor_date`);
- уменьшает дублирование (одни и те же факты, переписанные разными каналами);
- фильтрует шум с помощью лёгкого LLM-судьи, оставляя только полезные документы.

1.1 Команда

- **Артём Каменцев** спроектировал и реализовал ядро `TemporalRAG`: построение индекса (BM25 + эмбеддинги + FAISS), гибридный поиск (RRF), временное ранжирование, дедупликацию/кластеризацию кандидатов, интеграцию LLM-фильтра и финальную суммаризацию.
- **Борис Сажин** отвечал за сбор и предобработку данных, формирование датасета, подготовку инфраструктуры экспериментов и анализ результатов.

2 Related Work

В основе работы лежат современные подходы к извлечению информации и генерации: Retrieval-Augmented Generation (Lewis et al., 2020), sentence-transformer эмбеддинги семейства E5 (Wang et al., 2022), алгоритм Reciprocal Rank Fusion (Cormack et al., 2009) для гибридного поиска и LLM-as-a-Judge (Zheng et al., 2023) для автоматизированной оценки качества.

3 Dataset

В проекте использован собранный корпус новостных сообщений из Telegram:

- **Исходный объём:** 130 000 сообщений.

Структура датасета (ключевые поля):

- `message_id` — уникальный идентификатор сообщения;
- `id_channel` и `channel_name` — канал-источник (имена подтягиваются из таблицы каналов);
- `message` — текст сообщения;
- `date` — timestamp публикации;
- `date_day` — нормализованная дата (UTC, округление до дня), используется в темпоральных операциях;
- `topic` — топик, размеченный LLM по тексту новости;
- `channel_w` — вес канала на основе числа подписчиков (нормированный $\log(1 + \text{subs})$).

4 Approach

Реализация разделена на два основных компонента: построитель индекса и сам пайплайн ответа.

4.1 1. Индексирование: `TemporalRAGIndexBuilder`

`TemporalRAGIndexBuilder` подготавливает артефакты для гибридного поиска:

1. **Токенизация и BM25.** Тексты нормализуются и токенизируются, после чего строится `BM25Okapi`.
2. **Dense-эмбеддинги.** Документы кодируются через `SentenceTransformer` (по умолчанию `intfloat/multilingual-e5-large`) с префиксом `passage:` и L2-нормировкой.
3. **FAISS.** Для dense retrieval строится `faiss.IndexFlatIP` (скалярное произведение эквивалентно cosine similarity для нормированных векторов).
4. **Сохранение артефактов.** В папку индекса сохраняются `.npy` (эмбеддинги), `.index` (FAISS), `.pkl` (BM25 токены) и `rowmap.parquet` (маппинг строк на метаданные).

4.2 2. Hybrid Retrieval + Temporal RRF

Кандидаты для ответа выбираются гибридным поиском `hybrid_retrieve_rrf`:

- **Dense retrieval:** FAISS ищет `topN_each` ближайших документов по эмбеддингу запроса (префикс `query:`).
- **Sparse retrieval:** BM25 ранжирует весь корпус по лексической близости и выбирает `topN_each`.

- **Временная фильтрация:** Если задана актуальная дата, то в кандидаты допускаются только документы с датой `date_day` $\leq$ `anchor_date`. Дополнительно применяется окно истории `max_window_days` (по умолчанию 365): из прошлого берутся только документы, удовлетворяющие `anchor_date` $-$ `date_day` $\leq$ `max_window_days`, чтобы отсечь слишком далёкий контекст.

Слияние рангов происходит через взвешенный RRF, дополненный временным компонентом:

$$\text{score}(d) = \frac{w_{\text{dense}}}{k_{\text{rrf}} + \text{rank}_{\text{dense}}(d)} + \frac{w_{\text{bm25}}}{k_{\text{rrf}} + \text{rank}_{\text{bm25}}(d)} + \frac{w_{\text{time}}}{k_{\text{rrf}} + \text{rank}_{\text{time}}(d)}. \quad (1)$$

Здесь $\text{rank}_{\text{time}}(d)$ вычисляется по возрастанию age_days = `anchor_date` $-$ `date_day`: самые свежие документы получают ранг 1.

Опционально добавляется небольшая поправка за «вес канала» `channel_w`.

4.3 3. Dedup/Cluster: удаление семантических дублей

После retrieval кандидаты проходят постобработку `dedup_cluster_candidates_time`:

1. **Грубый дедуп.** Текст нормализуется (удаляются URL, @handle, лишняя пунктуация), затем добавляется дата.

2. **Кластеризация близких документов.** Для представителей групп считаются эмбеддинги и строится kNN-граф (FAISS/матрица сходства). Документы объединяются в кластеры, если cosine similarity ≥ 0.95 и разница по дате ≤ 1 день.

3. **Отбор.** Из каждого кластера оставляется `keep_per_cluster` документов с наивысшим score. А все каналы, которые писали об этом объединяются в список.

4.4 4. LLM Filtering (Judge)

Чтобы убрать остаточный шум, используется LLM-судья (`judge_filter_candidates`). Для каждого кандидата модель возвращает JSON:

```
{"relevance": 0|1|2}
```

и мы оставляем документы с релевантностью ≥ 1. Используется та же модель, что и для суммаризации, чтобы не выгружать дополнительную (показатели времени на 32b модели были удовлетворительными).

4.5 5. Generation

Финальное саммари строится LLM (`Qwen/Qwen2.5-32B-Instruct` через `vLLM`). Контекст формируется функцией `build_rag_context`:

- выбирается `k_docs` документов;
- приоритет отдаётся «актуальному окну» (по умолчанию 30 дней до `anchor_date`) с долей `hot_ratio`;
- документы сортируются от новых к старым и подаются в промпт в формате `date=YYYY-MM-DD channel(s)=...`.

5 Experiments

5.1 Выбор эмбеддера для dense retrieval

Перед основными экспериментами с retrieval мы выбрали эмбеддер для dense-компоненты (FAISS). Рассматривались несколько моделей, для которых заранее были посчитаны эмбеддинги документов (FP16/FP32), после чего поиск выполнялся по cosine similarity (через скалярное произведение нормированных L2-векторов). Качество retrieval оценивалось с помощью LLM-разметки релевантности в top-k по шкале $\{0, 1, 2\}$ (0 — нерелевантно, 1 — частично релевантно, 2 — строго релевантно), а затем считались метрики ранжирования и точности.

Скрининг нескольких эмбеддеров на top-10. На первом этапе мы провели быстрый отбор кандидатов по метрикам top-10, сравнив `e5-small`, `e5-base`, `e5-large`, `MiniLM` и `gte`. Сравнение показало, что при сопоставимом уровне P@10(rel ≥ 1) (наиболее высокий показатель давали более лёгкие модели), **E5-Large** лучше ранжирует выдачу и раньше поднимает строго релевантные документы: nDCG@10 = 0.952 и MRR@10(rel = 2) = 0.554. Близким конкурентом оказался **E5-Base** (nDCG@10 = 0.949, MRR@10(rel = 2) = 0.530), поэтому на следующем шаге мы сузили сравнение до пары Base vs Large.

Сравнение E5-Base и E5-Large на top-50. На глубине top-50 различия между Base и Large по средним retrieval-метрикам оказались небольшими: nDCG@50 практически совпал (0.6741 vs 0.6743), а по precision и MRR наблюдались лишь умеренные колебания. При раздельной агрегации по типам запросов виден разный профиль: для **topic**-запросов Large чуть лучше по nDCG и precision, тогда как для **news**-запросов отличия невелики и могут менять знак в зависимости от метрики. Таким образом, только по «классическим» retrieval-метрикам на top-50 уверенного доминирования одной из моделей не наблюдалось.

Качественный анализ семантического пространства и метрика уверенности (margin). Чтобы принять решение для production-пайплайна, мы дополнили сравнение анализом геометрии эмбеддингов. Во-первых, UMAP-проекция эмбеддингов документов показала, что **E5-Large** формирует более «структурированное» пространство с визуально более чёткими границами тематических кластеров, что снижает смешение близких, но разных тем. Во-вторых, мы измерили *margin* — разрыв между сходством лучшего совпадения и ближайшего «похожего» кандидата. На случайной выборке из 100 документов средний margin оказался выше у **E5-Large** (0.0764 против 0.0710 у Base), что интерпретируется как более уверенное отделение правильного совпадения от семантически близкого шума.

Итоговый выбор. С учётом (i) лучшего качества ранжирования на этапе top-10-скрининга, (ii) отсутствия деградации на top-50 и (iii) более высокой «уверенности» разделения по margin, для dense retrieval в дальнейшем был выбран **E5-Large**. В реализации эмбеддинги нормируются по L2 и используются в FAISS с inner product (эквивалент cosine similarity при нормировке).

5.2 Retrieval evaluation

Цель экспериментов по retrieval — сравнить базовые и гибридные ретриверы, а также оценить влияние временного компонента в ранжировании на «классические» метрики качества поиска. Мы использовали набор из **40 запросов** двух типов:

- **topic**: запрос задаёт тему (например, «курс доллара», «ключевая ставка», «нефть»);
- **news**: запрос формулируется как конкретное новостное событие/тезис (например, «ЦБ поднял курс доллара выше 100»).

Оцениваемые методы. Сравнивались четыре стратегии retrieval:

1. **BM25**: лексический поиск по токенизированному тексту.
2. **Semantic (FAISS)**: dense retrieval по эмбеддингам SentenceTransformer (cosine similarity через FAISS IndexFlatIP).
3. **Hybrid RRF**: объединение рангов dense и BM25 через Reciprocal Rank Fusion.
4. **Temporal RRF (temporal_rrf)**: Hybrid RRF, дополненный временным ранжированием относительно `anchor_date` (ранг по свежести) и временной фильтрацией (отсечение будущих документов и ограничение окна истории).

Разметка релевантности и метрики. Для вычисления retrieval-метрик нам нужна разметка релевантности документов для каждого запроса. В ручной разметке мы были ограничены по времени, поэтому использовали автоматическую разметку **LLM-судьёй** по дискретной шкале $\{0, 1, 2\}$:

- 0 — нерелевантно (не помогает ответить на запрос);
- 1 — частично релевантно (упоминает тему/контекст, но мало фактов);
- 2 — строго релевантно (содержит факты/цифры/событие по запросу).

Далее считались метрики на глубине $k = 50$:

- **nDCG@50** — качество ранжирования с учётом градаций релевантности;
- **MRR@50 (rel=2)** — насколько рано в выдаче появляется строго релевантный документ;
- **P@50 (rel=2)** — доля строго релевантных документов в топ-50;
- **P@50 (rel≥1)** — доля документов хотя бы частично релевантных в топ-50.

Результаты по всем запросам. Сводные результаты по всему набору запросов (оба типа вместе) приведены в Таблице 1. Hybrid RRF даёт лучший общий ранжирующий эффект (максимальные nDCG@50 и MRR@50), в то время как pure semantic retrieval приносит немного больше релевантных документов в топ-50 (precision), но хуже их упорядочивает.

Таблица 1: Качество retrieval по всем запросам (summary all), $k = 50$

Method	nDCG@50	MRR@50 (rel=2)	P@50 (rel=2)	P@50 (rel≥1)
BM25	0.739282	0.521766	0.195867	0.385082
Hybrid RRF	**0.783114**	**0.593155**	0.267857	0.469888
Semantic (FAISS)	0.763042	0.572750	**0.280347**	**0.484347**
Temporal RRF (ours)	0.773529	0.570238	0.265888	0.465388

Результаты по типам запросов. Чтобы понять поведение методов в разных режимах, мы также агрегировали метрики отдельно для **topic** и **news** (Таблица 2). Видно, что для **topic**-запросов semantic/hybrid дают максимально высокие метрики, тогда как **temporal_rrf** демонстрирует небольшое снижение — ожидаемое следствие того, что метод дополнительно оптимизирует свежесть контекста. Для **news**-запросов temporal_rrf остаётся конкурентоспособным и близок к лучшим значениям, при этом выигрывая по ряду показателей у чисто dense retrieval.

Таблица 2: Качество retrieval по типам запросов, $k = 50$

q_type	method	nDCG@50	MRR@50 (rel=2)	P@50 (rel=2)	P@50 (rel≥1)
news	bm25	0.650798	0.329167	0.036735	0.158163
news	hybrid	**0.660490**	0.287500	0.035714	**0.188776**
news	semantic	0.612732	0.239251	0.034694	0.184694
news	temporal_rrf	0.654856	**0.304167**	**0.038776**	**0.188776**
topic	bm25	0.827766	0.714365	0.355000	0.612000
topic	hybrid	0.905737	0.898810	0.500000	0.751000
topic	semantic	**0.913351**	**0.906250**	**0.526000**	**0.784000**
topic	temporal_rrf	0.892201	0.836310	0.493000	0.742000

Почему в основе temporal_rrf лежит hybrid. Метод **temporal_rrf** построен как расширение **hybrid RRF**: мы берём гибридное объединение dense (FAISS) и sparse (BM25) через Reciprocal Rank Fusion и добавляем к нему явную **временную составляющую** (ранг по свежести относительно `anchor_date` + временные ограничения на кандидатов). Такой выбор основы был сделан по результатам сравнения ретриверов: **hybrid** показывает почти максимальные метрики на **topic**-запросах (уступая semantic незначительно), но при этом работает **заметно стабильнее на news**-запросах и превосходит чистый semantic retrieval, который чаще «уходит» в семантически похожие, но нерелевантные/устаревшие формулировки.

Таким образом, гибридный поиск оказался лучшим компромиссом между лексической точностью (BM25) и семантическим покрытием (dense), поэтому именно его мы использовали как базу для temporal_rrf. Добавление временного компонента приводит лишь к **небольшому снижению** классических retrieval-метрик в среднем по запросам, что ожидаемо (метрики не оптимизируют свежесть), однако падение остаётся умеренным, а на части сценариев (в частности, для **news**-запросов) temporal_rrf сохраняет конкурентоспособность и местами даёт **лучшие значения** (например, по MRR@50(rel=2) и P@50(rel=2) на news). Это подтверждает, что внедрение времени улучшает практическую полезность выдачи, не «ломая» качество retrieval по стандартным IR-метрикам.

Фильтрация документов по релевантности перед саммари. Анализ результатов retrieval показал, что даже у сильных ретриверов в топ-k сохраняется заметная доля слабых кандидатов (оценка LLM-судьёй 0 или 1), особенно в сценариях **news**, где встречаются семантически похожие, но фактически бесполезные сообщения. Поэтому мы добавили отдельный этап **LLM-фильтрации релевантности** (judge) перед генерацией саммари: для каждого документа модель предсказывает $\{0, 1, 2\}$, после чего в контекст допускаются только документы с релевантностью ≥ 1.

Ожидалось, что такая фильтрация повысит **релевантность top-k контекста** (и уменьшит шум) независимо от выбранного ретривера, а эффект будет особенно заметен для гибридных и темпоральных вариантов, которые могут приносить больше разнообразных кандидатов. В итоге этот шаг делает контекст более «чистым» для генератора и напрямую улучшает качество итогового дайджеста по критериям релевантности и опоры на источники.

5.3 Summarization evaluation (LLM-as-a-judge)

Цель этой части — оценить не только «классическую» релевантность, но и соответствие ключевому требованию проекта: **актуальность относительно опорной даты** и корректная работа с временной осью новостей. Мы сравнили три системы на **200 запросах**:

1. **rag**: семантический retrieval + генерация без фильтрации;
2. **rag_filtered**: семантический retrieval + LLM-фильтр релевантности документов + генерация;

3. **temporal_filtered**: **Temporal RRF** (гибрид + временной компонент) + дедуп/кластеризация + LLM-фильтр + генерация.

Протокол оценки. Для каждого запроса формировались: **query**, **anchor**, **context** (набор документов с полями `date=YYYY-MM-DD channel=...`) и итоговый **summary**, сгенерированный системой. Далее независимый LLM-судья оценивал саммари **только по предоставленному CONTEXT** (внешние знания запрещены), не учитывая стиль и красоту текста — только содержательную полезность и проверяемость. Такой протокол важен для RAG-задачи, поскольку позволяет измерять качество именно как функцию ретривала/контекста и корректности опоры на источники.

Критерии (шкала 1..5). Мы использовали 5 критериев качества, каждый из которых отражает отдельный риск RAG-системы:

- **relevance** — тематическая релевантность запросу. Штрафуется явный оффтоп или уход в общие рассуждения без ответа на запрос.
- **groundedness** — привязка к источникам: насколько ключевые утверждения SUMMARY проверяемы по контексту и не содержат «додумок» или противоречий.
- **facts_numbers** — точность фактов и чисел относительно контекста (ошибки в цифрах и фактах критичны для финансового домена).
- **dates_timeline** — корректность дат и порядка событий: не путаются ли раньше/позже, не приписываются ли даты, которых нет в источниках.
- **recency** — Критерий актуальности. Судья определяет самые свежие документы как документы с максимальной датой в контексте (наиболее близкие к актуальной дате). Высокая оценка ставится только если саммари явно отражает «что нового» и использует факты из свежих документов (ориентир — последние 30 дней), а низкая — если свежие документы игнорируются или старое подаётся как новое.

Итоговая метрика overall. Чтобы получить единый показатель качества саммари, мы агрегировали пять оценок в **overall** как взвешенную сумму с приоритетом актуальности и корректного таймлайна:

$$\text{overall} = 0.45\cdot\text{recency}+0.20\cdot\text{dates\_timeline}+0.20\cdot\text{groundedness}+0.10\cdot\text{facts\_numbers}+0.05\cdot\text{relevance}. \tag{2}$$

Такой выбор весов отражает цель проекта: в новостном (особенно финансовом) сценарии критичнее всего не красиво пересказать, а **правильно выделить последнее** (recency), не перепутать последовательность событий (dates_timeline) и не делать утверждений без опоры на источники (groundedness). При этом factual- и topic-компоненты также учитываются, но с меньшим весом.

Формат ответа судьи и агрегация. Судья возвращал структурированный json со скорингом по каждому критерию (1..5), списком проблем (`issues`), выдержками-доказательствами (`evidence`: связка цитата из summary и подтверждающая цитата из контекста), а также `confidence` в диапазоне 0..1. Для финальной сводной метрики **overall** использовалась взвешенная сумма по 5 критериям с повышенным весом **recency** (и дополнительно учитывалась `confidence`), поскольку задача проекта — именно **производить актуальные дайджесты** и корректно работать со временем.

Почему temporal_filtered соответствует целям проекта. TemporalRAG оптимизирует цепочку «контекст → саммари» под требования новостного домена:

- **Учет времени на retrieval-этапе**: temporal_rrf добавляет временной ранг и отсечение документов из будущего относительно актуальной даты, тем самым повышая вероятность попадания свежих фактов в контекст.
- **Снижение дубликатов**: дедуп/кластеризация уменьшают долю семантических дублей (когда разные каналы переписывают один и тот же факт), освобождая место в контексте для новых событий.
- **Фильтрация шума**: LLM-фильтр релевантности перед генерацией отсеивает слабые документы и повышает среднюю полезность top-k контекста.

В результате temporal_filtered чаще формирует контекст, где присутствуют свежие, недублирующиеся факты, и генератору проще явно ответить «что нового» без галлюцинаций и без нарушения таймлайна. Это напрямую отражается в росте **recency** и **dates_timeline**, а также улучшает итоговый **overall** по сравнению с базовым RAG.

Результаты. .

Таблица 3: Свежесть дат в контексте и саммари (mean по запросам)

System	has_dates	n_dates	min_ctx	min_sum	GAP	sum_7d	sum_30d
rag	1.000	23.650	22.575	33.135	10.560	0.0211	0.0823
rag_filtered	0.990	25.475	30.788	33.838	3.051	0.0195	0.0686
temporal_filtered	0.995	21.525	**7.618**	**10.492**	**2.874**	**0.1471**	**0.3190**

Вывод по темпоральным метрикам. Ручные метрики подтверждают, что TemporalRAG решает главную проблему новостного RAG: контекст становится заметно свежее (больше документов в окнах 7/30 дней, меньший min/p50 age), а саммари чаще отражает именно актуальные события, уменьшая «отставание» от самого свежего доступного источника. Это согласуется с ростом критерия *recency* в LLM-as-a-judge оценке и демонстрирует соответствие требованиям проекта по работе с временем.

Таблица 4: Средние оценки саммари (200 запросов)

System	Relevance	Groundedness	Facts/Nums	Dates	Recency	Overall
rag	4.283	3.955	4.076	3.980	3.146	3.624
rag_filtered	4.379	4.000	4.121	4.091	3.303	3.736
temporal_filtered	4.317	4.005	4.236	4.206	**3.598**	**3.901**

6 Conclusion

Мы реализовали Temporal RAG — практичный пайплайн для работы с новостными потоками, где важна актуальность. Ключевые инженерные компоненты: гибридный поиск (FAISS+BM25) с RRF, явная временная поправка к рангу относительно `anchor_date`, дедупликация близких сообщений, фильтрация релевантности лёгкой LLM и генерация итогового дайджеста. Эксперименты показали улучшение итогового качества саммари и рост *recency* по сравнению с базовым RAG.