# synrag + Synapse — технический бенчмарк

**Synlabs · 26 ноября 2026 · версия 1.16**

synrag — движок поиска и подготовки контекста для LLM-агентов: документы → индекс → поиск → сжатый контекст с цитатами. Synapse — собственная многоязычная модель эмбеддингов, дистиллированная из EmbeddingGemma 2. Всё написано на Rust без внешних зависимостей, работает на обычном CPU.

В отчёте — полные технические параметры и сравнение на одной машине и одних данных с тем, что используют компании сегодня: Elasticsearch (BM25 и гибрид BM25 + векторы) и открытые модели эмбеддингов (all-MiniLM-L6-v2, bge-small-en-v1.5, multilingual-e5-small, EmbeddingGemma 2) — в стандартной и в оптимизированной настройке (ONNX Runtime int8 на CPU, fp16 на GPU), Model2Vec — самые быстрые открытые эмбеддинги для CPU (Python и официальная Rust-версия), и готовый open-source стек на Rust: **Tantivy (BM25) + Model2Vec-RS + гибридное слияние** — то, что инженер может собрать сам вместо synrag. Отдельно — качество ответа LLM и экономия токенов. Все цифры — наши замеры; методика и скрипты в репозитории.

## 1. Главное

| | Результат |
|---|---|
| **Масштаб** | MS MARCO, **8.84 млн текстовых отрывков** (в среднем ~340 байт, ~55 слов; не полноразмерные PDF или офисные документы) на одном 6-ядерном сервере без GPU: BM25 готов за 22 с, BM25 + Synapse — за 2.4 мин; nDCG@10 0.244 / **0.277** (опубликованный BM25 — 0.228); запрос p50 7–10 мс; пик памяти 15–16 ГБ |
| **Скорость создания базы** | 57.6 тыс. документов (FiQA, 44 МБ текста) от запуска процесса до готовности к поиску, CPU: **1.6 с** (BM25 + Synapse) или 4.8 с (+ модель корпуса). Elasticsearch: 12.7 с (запуск 9.5 + индекс 3.2). Модели эмбеддингов на GPU: 60–282 с; на том же CPU: 16 мин – 1.7 ч |
| **Скорость эмбеддингов** | Synapse: **51.6 тыс. док/с на 6-ядерном CPU**. Трансформерные модели на том же CPU: 9–58 док/с, с ONNX int8 66–125 док/с; на GPU RTX 5060 Ti: 150–1375 док/с, с fp16 1765–2252 док/с. Статический Synapse (турбо-режим): **~235 тыс. док/с**, качество −0.01. Model2Vec-RS (Rust): многоязычная модель — 56 тыс. док/с, английская — 750 тыс. док/с |
| **Качество поиска** | режимы указаны явно — «быстрый»: база FiQA за 1.6 с, «полный»: за 4.8 с. Английский (3 набора BEIR): synrag полный 0.444, полный + переранжировщик **0.455** nDCG@10 (быстрый + переранжировщик 0.451) — выше Elasticsearch BM25 (0.424) и all-MiniLM-L6-v2 (0.445), на уровне гибридов Elasticsearch + модель на GPU (0.455–0.474), ниже bge-small (0.478) и EmbeddingGemma (0.503). 5 языков (MLQA): synrag полный 0.653, полный + переранжировщик **0.664** (быстрый + переранжировщик 0.660) — на уровне лучших и чуть выше (+0.006…+0.007, значимость не проверялась): EmbeddingGemma на GPU (0.658), гибрида Elasticsearch + EmbeddingGemma на GPU (0.657), Elasticsearch + e5-small (0.641), стека Tantivy + Model2Vec-RS (0.644), Elasticsearch (0.585). 12 языков MQA, включая турецкий, казахский, узбекский, азербайджанский и русский: synrag быстрый 0.460, быстрый + переранжировщик **0.478** (полный + переранжировщик 0.474) — выше Elasticsearch (0.416), e5-small (0.409), их гибрида (0.439), гибрида Elasticsearch + EmbeddingGemma (0.445) и Tantivy + Model2Vec-RS (0.427), ниже одной EmbeddingGemma на GPU (0.486) |
| **Ресурсы** | память при работе 61–475 МБ против 4.9 ГБ у Elasticsearch и 1.6–2.8 ГБ + 0.7–5.6 ГБ видеопамяти у моделей эмбеддингов. Программа 1.75 МБ, модель 86 МБ. GPU не нужен |
| **Обновление базы** | без перестройки и без остановки поиска (FiQA, 57.6 тыс. документов, с Synapse): добавить или заменить документ — **0.2 мс**, удалить — **< 0.05 мс**, 1000 документов — 30 мс; поиск в 4 потока при ~1100 обновлениях/с — p50 0.87 мс; качество после обновлений равно полной перестройке; журнал изменений — восстановление после сбоя без потери обновлений |
| **Задержка запроса** | по HTTP, одним клиентом (FiQA): synrag быстрый p50 **1.35 мс** / p99 2.05 мс и 4.1 тыс. запр/с при 171 МБ памяти против Elasticsearch 1.96 / 4.34 мс и 3.2 тыс. запр/с при 4.7 ГБ; с переранжировщиком 3.2 мс. Внутри процесса synrag — 0.03–0.9 мс; модели эмбеддингов — 3–25 мс (кодирование запроса + поиск) |
| **Стоимость индексации** | 1 млн документов размера FiQA в готовую базу на облачном CPU (AWS c7a.2xlarge, $0.41/ч): **~$0.003** (28 с). Только векторы: e5-small на GPU L4 — $0.12, EmbeddingGemma на GPU — $1.08, API OpenAI / Titan — $3.8 (раздел 8г) |
| **Токены LLM** | Qwen3-8B, 2100 вопросов, 7 языков: сопоставимое качество ответа при **на 43 % меньшем промпте** (357 против 624 токенов), чем у обычного RAG (склейки найденных документов); разница F1 +0.000 [−0.015; +0.017]. Это экономия входных токенов на один вызов LLM, а не полная стоимость работы агента |
| **Enterprise** (версия 1.9) | коллекции, ключи API и SSO-токены (OIDC: Keycloak, Entra ID, Okta, Google), роли, **права на каждый документ** (фильтр внутри поиска: +0.24 мс), журнал аудита с цепочкой хэшей, шифрование хранения, метрики Prometheus, кластер из шардов и реплик, коннекторы (S3, Confluence, SharePoint, Google Drive, SQL), Docker-образ 27 МБ, Helm-чарт. По HTTP с ключом: p50 **0.78 мс**, 6.0 тыс. запр/с, качество то же, что без авторизации; 2 шарда: nDCG −0.001 (раздел 8д) |

| | |
|---|---|
| **Независимые наборы** | данные, не участвовавшие ни в обучении, ни в настройке (раздел 7в). 14 языков Belebele: synrag быстрый + переранжировщик **0.874** — выше EmbeddingGemma на GPU (0.852), гибрида Elasticsearch + EmbeddingGemma (0.853) и Elasticsearch (0.791); отрыв дают казахский, узбекский, киргизский, таджикский (+0.10…+0.24), на 8 языках из 14 впереди Gemma. 4 английских набора BEIR (SciDocs, ArguAna, TREC-COVID, Quora): synrag 0.518 — выше Elasticsearch (0.503), на уровне e5-small и одной Gemma (0.514 / 0.524), **ниже bge-small (0.557) и гибридов Elasticsearch + модель на GPU (0.537–0.561)** |

**Вывод.** synrag даёт качество поиска на уровне небольших открытых моделей эмбеддингов, выше Elasticsearch BM25 и выше связки Elasticsearch + Model2Vec. Против трансформерных моделей у synrag разрыв в скорости на порядки. База готова к поиску за 1.6 с на CPU: в **36–170 раз быстрее**, чем модели эмбеддингов на GPU в стандартной настройке (в **17–22 раза** — с fp16), и в **280–530 раз быстрее** их лучших CPU-вариантов (ONNX int8). Памяти в 4–10 раз меньше, GPU не нужен. Разрыв по скорости — следствие архитектуры (в тысячи раз меньше операций на токен), а не настроек: оптимизации конкурентов дают им ×1.3–2 (раздел 5). По чистому качеству на английском лучшие модели эмбеддингов на GPU остаются впереди: bge-small на 0.023, EmbeddingGemma на 0.048 nDCG@10 (против synrag полный + переранжировщик). На пяти языках MLQA synrag на уровне EmbeddingGemma на GPU (полный + переранжировщик 0.664, быстрый + переранжировщик 0.660, Gemma 0.658). На 12 языках MQA EmbeddingGemma в среднем немного впереди (0.486 против 0.478 у быстрого + переранжировщика): synrag лучше на казахском, узбекском, азербайджанском, урду и немецком, Gemma — на русском, турецком, английском, арабском, персидском и хинди. На независимых данных (раздел 7в) картина та же: на 14 языках Belebele synrag в среднем выше EmbeddingGemma (0.874 против 0.852) благодаря казахскому, узбекскому, киргизскому и таджикскому, а на четырёх новых английских наборах BEIR уступает bge-small и гибридам Elasticsearch + модель на GPU (0.518 против 0.537–0.561).

**Переранжировщик (версия 1.7)** — контекстная модель на 66 тыс. параметров поверх Synapse: для первых 30 кандидатов каждое слово вопроса ищет лучшее место в документе. Обучен повторять ранжирование учителя (EmbeddingGemma). Прибавляет **+0.010…+0.011 nDCG@10 на английском, +0.009…+0.011 на пяти языках, +0.018 на 12 языках СНГ и Ближнего Востока** (раздел 6–7б). Построение базы не меняется; запрос — ~3 мс вместо ~0.6 мс, пропускная способность ниже в ~24 раза (~11.5 мс CPU на запрос, раздел 8). Включается флагом `--rerank`.

**Главный конкурент — готовый стек на Rust: Tantivy + Model2Vec-RS** (раздел 7а). Он так же работает на CPU, стартует так же быстро или быстрее (FiQA: 0.44 с с английской моделью, 2.8 с с многоязычной; synrag — 1.6 с) и уступает по качеству: synrag лучше на английском (BEIR +0.005, MLQA en +0.027), на пяти языках MLQA (+0.007) и на 12 языках MQA, включая тюркские (+0.033; один BM25 synrag — +0.025 к BM25 Tantivy). С переранжировщиком отрыв растёт: BEIR +0.015…+0.021, MLQA +0.016…+0.020, MQA +0.051. До стеммеров версий 1.3–1.4 synrag отставал на пяти языках на 0.010. Synapse как энкодер не быстрее Model2Vec. Преимущество synrag перед этим стеком — не скорость эмбеддингов, а готовый продукт: парсеры документов, сжатый контекст с цитатами, HTTP-сервер, один файл; и меньше памяти, чем с многоязычной моделью Model2Vec (443 МБ против 1.85 ГБ).

## 1а. Сводная таблица: все против всех

Качество — nDCG@10 на test; скорость и ресурсы — FiQA (57.6 тыс. документов), та же машина (6 ядер, 31 ГБ ОЗУ, RTX 5060 Ti 16 ГБ). «Готов к поиску» — от запуска до готовности. Кодирование моделей — лучшая настройка (fp16 на GPU / ONNX int8 на CPU). Задержка моделей — кодирование запроса + точный поиск (Python, GPU).

| Система | GPU | Английский (BEIR) | 5 языков (MLQA) | 12 языков (MQA) | Готов к поиску | Кодирование, док/с | Запрос p50 | ОЗУ | Видеопамять |
|---|---|---|---|---|---|---|---|---|---|
| **synrag быстрый** | нет | 0.441 | 0.651 | 0.460 | **1.6 с** | 51.6 тыс. | **0.55 мс** | 443 МБ | — |
| **synrag быстрый + переранжировщик** | нет | 0.451 | 0.660 | 0.478 | 1.6 с | 51.6 тыс. | 2.9 мс | 443 МБ | — |
| **synrag полный + переранжировщик** | нет | 0.455 | **0.664** | 0.474 | 4.8 с | 51.6 тыс. | ~3 мс | 475 МБ | — |
| synrag турбо | нет | 0.429 | 0.643 | 0.453 | 0.76 с | 235 тыс. | ~0.6 мс | ~443 МБ | — |
| Elasticsearch BM25 | нет | 0.424 | 0.585 | 0.416 | 12.7 с | — | 2.2–4.2 мс | 4.9 ГБ | — |
| Tantivy + Model2Vec-RS, многоязычная | нет | 0.430 | 0.644 | 0.427 | 2.8 с | 56 тыс. | 3.8 мс | 1.85 ГБ | — |
| Tantivy + Model2Vec-RS, английская | нет | 0.436 | 0.630 | — | 0.44 с | 750 тыс. | 3.7 мс | 370 МБ | — |
| all-MiniLM-L6-v2 | да | 0.445 | 0.198 | 0.203 | 60 с (CPU 16.5 мин) | 2252 (CPU 125) | ~3 мс | 2.1 ГБ | 0.7 ГБ |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| bge-small-en-v1.5 | да | 0.478 | 0.224 | 0.267 | 107 с (CPU 31 мин) | 1765 (CPU 66) | ~5 мс | 2.1 ГБ | 0.8 ГБ |
| multilingual-e5-small | да | 0.430 | 0.624 | 0.409 | 74 с (CPU 19 мин) | 1818 (CPU 71) | ~5 мс | 2.5 ГБ | 1.1 ГБ |
| EmbeddingGemma 2 | да | **0.503** | 0.658 | **0.486** | 282 с (CPU 1.7 ч) | 207 (CPU 9) | ~25 мс | 2.8 ГБ | 5.6 ГБ |
| Elasticsearch + e5-small / EmbeddingGemma | да | 0.455–0.474 (4 модели) | 0.641 / 0.657 | 0.439 / 0.445 | Elasticsearch + модель | как у модели | +2–4 мс | 4.9 ГБ + модель | как у модели |

- **Английский**: впереди EmbeddingGemma (0.503) и bge-small (0.478); synrag полный + переранжировщик (0.455) — на уровне нижней границы гибридов Elasticsearch + модель на GPU, выше MiniLM, e5-small, Elasticsearch и Tantivy + Model2Vec.
- **5 языков**: synrag, EmbeddingGemma и гибрид Elasticsearch + EmbeddingGemma в пределах 0.007 друг от друга.
- **12 языков, включая тюркские**: EmbeddingGemma 0.486, synrag 0.478, остальные 0.409–0.445. По языкам synrag первый на казахском, узбекском, азербайджанском, урду и немецком (раздел 7б).
- **Скорость и ресурсы**: synrag на CPU готов к поиску в 37–176 раз быстрее моделей на GPU (в сотни-тысячи раз — на CPU), отвечает так же или до 45 раз быстрее (0.55–2.9 мс против 3–25 мс), памяти в 6–20 раз меньше, чем у Elasticsearch и моделей эмбеддингов, видеопамять не нужна. Tantivy + английская Model2Vec легче (370 МБ) и стартует быстрее (0.44 с), но слабее по качеству вне английского.
- **Независимые наборы** (раздел 7в): Belebele, 14 языков — synrag 0.874, EmbeddingGemma 0.852 (перевес — казахский, узбекский, киргизский, таджикский; на 8 языках из 14 впереди Gemma); ещё 4 английских набора BEIR — synrag 0.518, bge-small 0.557, гибриды Elasticsearch + модель 0.537–0.561, Elasticsearch 0.503.
- **Ответ LLM** (раздел 9): промпт на 43 % короче при том же F1, чем у обычного RAG; F1 на +0.05…+0.07 выше, чем у отбора предложений по эмбеддингам (как LlamaIndex / LangChain).

## 2. Что внутри

| Компонент | Параметры |
|---|---|
| Программа `synrag` | 1.75 МБ, один исполняемый файл, 0 внешних зависимостей (Rust, только стандартная библиотека). CLI, HTTP-сервер, библиотека |
| Парсеры | PDF, DOCX, XLSX, PPTX, ODF, HTML, Markdown, CSV, JSON, ZIP — собственные. 994 реальных файла (678 МБ) → индекс за 2.7 с |
| Канал 1: BM25 | собственный анализатор для любого языка: стеммеры EN/RU, лёгкие стеммеры для арабского и хинди, обрезка основ до 6 букв (латиница, кириллица — в том числе тюркские языки), символьные n-граммы для китайского/японского/тайского |
| Канал 2: модель корпуса | учится на документах заказчика за ~3 с без разметки (2.6–26 млн параметров, int8) |
| Канал 3: Synapse | 21.3 млн параметров; свой токенизатор BPE (80 тыс. токенов); свёрточный кодировщик (окно ±8) + глобальный гейт; вектор 256, хранение int8. Файл 86 МБ, в памяти ~24 МБ. 129 языков и письменностей, включая казахский и узбекский |
| Переранжировщик | 65.8 тыс. параметров (файл 263 КБ) поверх токенов Synapse: проекции 128 × 256 для вопроса и документа, каждое слово вопроса берёт лучшее совпадение в документе (MaxSim), слова вопроса взвешены по важности. Работает на первых 30 кандидатах. Обучен за 38 мин на CPU повторять ранжирование учителя на 1.07 млн списков кандидатов BM25 (MQA, GooAQ, arXiv) |
| Учитель Synapse | EmbeddingGemma 2 (Apache-2.0), 271 млн параметров текстовой части; ~18 млн текстов разметки; обучение 60 тыс. шагов на CPU (~77 мин) |
| Python SDK | `python/`: клиент без зависимостей — запуск или подключение к серверу, поиск, контекст с цитатами, добавление и удаление документов, инструмент для агента в формате OpenAI (`tool()` / `call_tool()`); пример агента для любого OpenAI-совместимого LLM (OpenAI, Ollama, vLLM). Проверено с Qwen3-8B: вызов инструмента 2–3 мс |
| Сжатый контекст | отбор предложений под бюджет токенов, цитаты `[1] Документ — Раздел (p. 3)`, JSON-список источников; 0.3–1.8 мс |
| Сервер (версия 1.9) | коллекции, ключи API и SSO (JWT RS256/HS256), роли, права на документы, аудит, шифрование хранения (ChaCha20-Poly1305), метрики Prometheus, JSON-логи, пул потоков с отказом 503 при перегрузке, keep-alive, загрузка файлов (`/upload`) — раздел 8д и `docs/ENTERPRISE.md` |
| Кластер (версия 1.9) | координатор: шарды по хэшу источника, реплики с переключением при сбое, слияние оценок шардов; Helm-чарт (шарды × реплики, ingress с TLS) |

| | |
|---|---|
| Коннекторы (версия 1.9) | `python/synrag/connectors`: S3 (подпись SigV4), Confluence, SharePoint / OneDrive (Microsoft Graph), Google Drive, SQL (DB-API), папки; инкрементальная синхронизация, перенос прав доступа |

## 3. Методика

- **Машина** (одна для всех систем): AMD Ryzen 5 7500F (6 ядер / 12 потоков), 31 ГБ ОЗУ, NVIDIA RTX 5060 Ti 16 ГБ.
- **Данные**: BEIR SciFact (5.2 тыс. документов), NFCorpus (3.6 тыс.), FiQA (57.6 тыс.) — английский; MLQA (вопрос и абзацы Википедии на одном языке): английский (10.9 тыс.), китайский (5.0 тыс.), немецкий (4.5 тыс.), арабский (5.1 тыс.), хинди (4.5 тыс.). Везде **test**-выборки.
- **Честность выбора**: все настройки synrag выбирались только на train/dev; test использован для итоговых цифр.
- **Elasticsearch 8.15**: Docker, heap 4 ГБ, `english` analyzer, `multi_match` по title + text — стандартная постановка BEIR. Для не-английских языков тот же analyzer (с языковыми плагинами Elasticsearch может быть лучше).
- **Модели эмбеддингов**: sentence-transformers 6.3, PyTorch 2.11 + CUDA 12.8, fp32, батч 128, до 256 токенов, точный поиск (полный перебор, как FAISS Flat). Скорость на CPU — по 2000 документам FiQA, пересчитана на корпус.
- **Оптимизированные модели эмбеддингов** (отдельный замер, раздел 5): ONNX Runtime 1.23 на CPU — fp32 и int8 (динамическое квантование по каналам, инструкции AVX-512 VNNI); PyTorch fp16 на GPU. Те же тексты, префиксы и длина 256 токенов; батчи отсортированы по длине. EmbeddingGemma в оптимизированных вариантах не мерили.
- **Model2Vec 0.10**: potion-base-8M (английская) и potion-multilingual-128M — статические эмбеддинги на CPU, до 256 токенов, точный поиск.
- **Synapse**: таблица токенов хранится в int8, вычисления идут в f32. Полностью f32-версия даёт те же nDCG@10 (разница ≤ 0.0004 на пяти языках MLQA) и ту же скорость: квантование экономит память, но не даёт скорость.
- **Гибрид Elasticsearch + векторы**: BM25 Elasticsearch + косинус модели, min-max нормализация, вес 0.6 — так же, как synrag сливает свои каналы. Вес под каждую модель не подбирался.
- **Метрики**: nDCG@10 и Recall@100 по одной формуле для всех систем.

## 4. От холодного старта до готовой базы

FiQA: 57 638 документов, 44 МБ текста. «Готова» = можно искать. Для synrag время замерено внутри программы от запуска процесса до готовности (чтение корпуса, все каналы индекса, загрузка и прогон Synapse; без запросов), медиана 3 запусков. Этапы synrag выполняются последовательно.

| Система | Железо | Время до готовой базы | Память | Качество (nDCG@10) |
|---|---|---|---|---|
| **synrag BM25** | CPU | **0.45 с** (чтение 0.3 с + индекс 0.15 с) | 325 МБ | 0.261 |
| **synrag быстрый** (BM25 + Synapse) | CPU | **1.6 с** (+ загрузка и кодирование Synapse 1.2 с) | 443 МБ | 0.279 |
| **synrag турбо** (BM25 + статический Synapse) | CPU | **0.76 с** (кодирование 0.25 с) | — | 0.270 |
| **synrag полный** (BM25 + модель корпуса + Synapse) | CPU | **4.8 с** (+ обучение модели корпуса 3.1 с) | 475 МБ | 0.277 |
| Elasticsearch BM25 | CPU | 9.5 с запуск + **3.2 с** индекс | 4.9 ГБ | 0.254 |
| Model2Vec potion-base-8M | CPU | 0.6 с + **2.1 с** | 0.8 ГБ | 0.160 |
| Model2Vec potion-multilingual-128M | CPU | 3.2 с + **3.1 с** | 2.5 ГБ | 0.100 |
| **Tantivy + Model2Vec-RS** potion-base-8M (Rust) | CPU | **0.44 с** (BM25 0.2 с, модель 0.03 с, кодирование 0.08 с) | 370 МБ | 0.267 |
| **Tantivy + Model2Vec-RS** potion-multilingual-128M (Rust) | CPU | **2.8 с** (загрузка модели 1.45 с, кодирование 1.0 с) | 1.85 ГБ | 0.260 |
| all-MiniLM-L6-v2 | GPU | 1.7 с загрузка + **58 с** | 2.1 ГБ + 0.7 ГБ GPU | 0.369 |
| all-MiniLM-L6-v2 | CPU | **~16.5 мин** | 1.7 ГБ | 0.369 |
| all-MiniLM-L6-v2, оптимизированный | GPU fp16 / CPU ONNX int8 | **~27 с** / **~7.7 мин** | — | 0.369 |
| bge-small-en-v1.5 | GPU | 1.8 с + **105 с** | 2.1 ГБ + 0.8 ГБ GPU | 0.390 |
| bge-small-en-v1.5 | CPU | **~31 мин** | 1.7 ГБ | 0.390 |

| | | | | |
|---|---|---|---|---|
| multilingual-e5-small | GPU | 4.3 с + **70 с** | 2.5 ГБ + 1.1 ГБ GPU | 0.334 |
| multilingual-e5-small | CPU | **~19 мин** | 2.3 ГБ | 0.334 |
| multilingual-e5-small, оптимизированный | GPU fp16 / CPU ONNX int8 | **~36 с** / **~13.5 мин** | — | 0.334 |
| EmbeddingGemma 2 | GPU | 4.3 с + **278 с** | 2.8 ГБ + 5.6 ГБ GPU | 0.308 |
| EmbeddingGemma 2 | CPU | **~1.7 ч** | 6.1 ГБ | 0.308 |
| Гибрид Elasticsearch + модель | GPU | время Elasticsearch + время модели | сумма | 0.306–0.331 |
| Гибрид Elasticsearch + Model2Vec | CPU | время Elasticsearch + 2.7–6.3 с | сумма | 0.259–0.269 |

FiQA (финансовые вопросы-ответы) — самый трудный для BM25 набор: здесь модели эмбеддингов заметно лучше synrag. Средние по всем наборам — в разделах 6–7.

## 5. Скорость создания эмбеддингов

Документы FiQA, до 256 токенов. «Стандарт» — sentence-transformers fp32; «оптимизированный» — ONNX Runtime int8 на CPU и fp16 на GPU (лучший из вариантов; ONNX fp32 на CPU не быстрее PyTorch).

| Модель | Параметры | Умножений на токен | CPU, стандарт | CPU, ONNX int8 | GPU, стандарт | GPU, fp16 |
|---|---|---|---|---|---|---|
| **Synapse (synrag)** | **21.3 млн** | **~5 тыс.** | **51 600 док/с** | — | не нужен | — |
| Model2Vec potion-base-8M, Python / **Rust** | 7.6 млн | ~0 (сумма векторов слов) | 27 600 / **750 000 док/с** | — | — | — |
| Model2Vec potion-multilingual-128M, Python / **Rust** | 128 млн | ~0 | 18 700 / **56 400 док/с** | — | — | — |
| all-MiniLM-L6-v2 | 22.7 млн | ~12 млн | 58 док/с | 125 док/с | 610–1375 док/с | 2252 док/с |
| bge-small-en-v1.5 | 33.4 млн | ~21 млн | 31 док/с | 66 док/с | 346–636 док/с | 1765 док/с |
| multilingual-e5-small | 117.7 млн | ~21 млн | 50 док/с | 71 док/с | 567–889 док/с | 1818 док/с |
| EmbeddingGemma 2 | 744 млн (271 млн текст) | ~100 млн | 9 док/с | не мерили | 150–207 док/с | не мерили |

- Оптимизации ускоряют трансформерные модели в 1.3–2 раза, качество почти не меняется (int8: SciFact −0.0002…−0.006 nDCG@10; fp16: без изменений). Они делают те же операции быстрее, но не уменьшают их число.
- Synapse делает на токен в 2 000–20 000 раз меньше умножений (свёртка по окну ±8 вместо слоёв внимания и полносвязных слоёв). Поэтому даже против оптимизированных вариантов он быстрее в **~410–780 раз** на том же CPU и в **~23–29 раз** против их fp16 на GPU.
- Model2Vec считает ещё меньше (только складывает готовые векторы слов). В официальной Rust-версии многоязычная модель кодирует с той же скоростью, что Synapse, а английская — в 13 раз быстрее. В Python-версии Model2Vec медленнее в 2–3 раза, но сравнивать надо с Rust-версией.
- **Статический Synapse** (версия 1.6, `--student-static`): отдельно обученный ученик без контекста (каждый токен — строка таблицы, текст — взвешенная сумма строк). FiQA: 57.6 тыс. документов за 0.245 с (**~235 тыс. док/с**, в 4.2 раза быстрее обычного Synapse и многоязычного Model2Vec-RS; английский Model2Vec-RS быстрее ещё в 3 раза). Качество в связке с BM25 (test): BEIR 0.429 (обычный 0.441), MLQA 0.643 (0.651), MQA 12 языков 0.453 (0.460; Tantivy + Model2Vec-RS 0.427). Отдельно, без BM25, заметно слабее (BEIR 0.262 против 0.319).

Что это значит на объёмах (**экстраполяция только скорости кодирования**; хранение, поиск и обновление на таких объёмах не проверялись, см. раздел 11):

| Корпус | Synapse, 1 сервер CPU | Model2Vec-RS multilingual, CPU | e5-small, GPU fp16 | e5-small, CPU ONNX int8 | EmbeddingGemma, GPU |
|---|---|---|---|---|---|
| 1 млн документов размера FiQA (~760 байт) | **~20 с** | ~18 с | ~9 мин | ~4 ч | ~1.4 ч |
| 100 млн таких документов | **~32 мин** | ~30 мин | ~15 ч | ~16 сут | ~5.8 сут |

## 6. Качество поиска: английский (BEIR test, nDCG@10)

| Система | SciFact | NFCorpus | FiQA | Среднее | GPU |
|---|---|---|---|---|---|
| Elasticsearch BM25 | 0.691 | 0.327 | 0.254 | 0.424 | нет |
| synrag BM25 | 0.695 | 0.317 | 0.261 | 0.424 | нет |
| **synrag быстрый** (BM25 + Synapse) | 0.706 | 0.338 | 0.279 | **0.441** | нет |
| **synrag полный** | 0.708 | 0.349 | 0.277 | **0.444** | нет |
| **synrag быстрый + переранжировщик** | 0.714 | 0.342 | 0.296 | **0.451** | нет |
| **synrag полный + переранжировщик** | 0.719 | 0.350 | 0.298 | **0.455** | нет |
| all-MiniLM-L6-v2 | 0.648 | 0.318 | 0.369 | 0.445 | да |
| multilingual-e5-small | 0.651 | 0.306 | 0.334 | 0.430 | да |
| bge-small-en-v1.5 | 0.703 | 0.343 | 0.390 | 0.478 | да |
| EmbeddingGemma 2 | 0.854 | 0.345 | 0.308 | 0.503 | да |
| Гибрид Elasticsearch + MiniLM / e5 / bge / Gemma | 0.716–0.753 | 0.344–0.361 | 0.306–0.331 | 0.455–0.474 | да |
| Synapse отдельно (без BM25) | 0.526 | 0.247 | 0.184 | 0.319 | нет |
| Model2Vec potion-base-8M | 0.523 | 0.249 | 0.160 | 0.310 | нет |
| Model2Vec potion-multilingual-128M | 0.444 | 0.172 | 0.100 | 0.239 | нет |
| Гибрид Elasticsearch + Model2Vec (8M / multilingual) | 0.699 / 0.695 | 0.336 / 0.332 | 0.269 / 0.259 | 0.435 / 0.429 | нет |

synrag полный на английском: на уровне MiniLM, выше e5-small и Elasticsearch, ниже bge-small (−0.034) и EmbeddingGemma (−0.059). Слабое место — FiQA (перефразированные финансовые вопросы).

Сравнение «система против системы»: гибриды Elasticsearch + модель на GPU лучше synrag на 0.011–0.030; связку Elasticsearch + Model2Vec на CPU synrag обходит (+0.006…+0.015). Synapse сам по себе слабее трансформерных моделей (0.319 против 0.43–0.50) и на уровне Model2Vec: он рассчитан на работу в паре с BM25. Переранжировщик (версия 1.7) прибавляет +0.010…+0.011, больше всего на FiQA (+0.018…+0.021) — там, где вопрос и ответ сформулированы разными словами. С ним synrag полный выше MiniLM (+0.010) и на нижней границе гибридов Elasticsearch + модель на GPU (0.455–0.474); bge-small и EmbeddingGemma впереди на 0.023 и 0.048. «synrag быстрый» и «полный» отличаются на 0.003 на test и на 0.005 на dev, поэтому быстрый режим — разумный выбор по умолчанию (модель корпуса стоит +3 с на старте).

## 7. Качество поиска: другие языки (MLQA test, nDCG@10)

| Система | английский | китайский | немецкий | арабский | хинди | Среднее | GPU |
|---|---|---|---|---|---|---|---|
| Elasticsearch BM25 | 0.711 | 0.622 | 0.573 | 0.482 | 0.538 | 0.585 | нет |
| synrag BM25 | 0.752 | 0.689 | 0.604 | 0.569 | 0.598 | 0.642 | нет |
| **synrag быстрый** (BM25 + Synapse) | 0.762 | 0.696 | 0.611 | 0.578 | 0.607 | **0.651** | нет |
| **synrag полный** | 0.750 | 0.697 | 0.617 | 0.587 | 0.614 | **0.653** | нет |
| **synrag быстрый + переранжировщик** | 0.777 | 0.704 | 0.620 | 0.580 | 0.617 | **0.660** | нет |
| **synrag полный + переранжировщик** | 0.767 | 0.707 | 0.627 | 0.593 | 0.625 | **0.664** | нет |
| all-MiniLM-L6-v2 | 0.586 | 0.063 | 0.317 | 0.007 | 0.015 | 0.198 | да |
| bge-small-en-v1.5 | 0.570 | 0.138 | 0.352 | 0.025 | 0.036 | 0.224 | да |
| multilingual-e5-small | 0.652 | 0.643 | 0.632 | 0.568 | 0.625 | 0.624 | да |
| EmbeddingGemma 2 | 0.660 | 0.682 | 0.677 | 0.594 | 0.676 | 0.658 | да |
| Гибрид Elasticsearch + e5-small | 0.747 | 0.684 | 0.628 | 0.541 | 0.606 | 0.641 | да |
| Гибрид Elasticsearch + Gemma | 0.756 | 0.707 | 0.641 | 0.557 | 0.625 | 0.657 | да |
| Synapse отдельно (без BM25) | 0.502 | 0.376 | 0.291 | 0.247 | 0.293 | 0.342 | нет |
| Model2Vec potion-base-8M | 0.521 | 0.107 | 0.168 | 0.042 | 0.032 | 0.174 | нет |
| Model2Vec potion-multilingual-128M | 0.424 | 0.447 | 0.373 | 0.323 | 0.364 | 0.386 | нет |
| Гибрид Elasticsearch + Model2Vec multilingual | 0.720 | 0.644 | 0.599 | 0.507 | 0.562 | 0.606 | нет |

- Популярные английские модели (MiniLM, bge-small) на других языках не работают (0.01–0.35).
- С переранжировщиком (версия 1.7) synrag на CPU чуть выше всех сравнённых систем (значимость разницы не проверялась): EmbeddingGemma на GPU (+0.006), гибрида Elasticsearch + EmbeddingGemma (+0.007), multilingual-e5-small (+0.040). По языкам впереди моделей остаются хинди (EmbeddingGemma 0.676 против 0.625) и немецкий (0.677 против 0.627).
- synrag на CPU выше multilingual-e5-small (+0.029) и гибрида Elasticsearch + e5-small на GPU (+0.012), на уровне EmbeddingGemma и гибрида Elasticsearch + EmbeddingGemma (−0.004…−0.005). Лучше всех на английском MLQA (0.750–0.762).
- Арабский — на уровне моделей эмбеддингов (0.587 против 0.568–0.594); хинди — у моделей сильнее (+0.011…+0.062).
- Версия 1.3: лёгкие стеммеры дали арабскому +0.061…+0.073, хинди +0.019…+0.022 (раньше: ар. 0.496–0.523, хинди 0.577–0.592); остальные языки не изменились.
- Связку Elasticsearch + Model2Vec (Python) synrag обходит на 0.044; со стеком Tantivy + Model2Vec-RS — см. раздел 7а. Synapse отдельно уступает potion-multilingual-128M (0.342 против 0.386), но в связке с BM25 synrag сильнее.

### 7а. synrag против готового стека на Rust: Tantivy + Model2Vec-RS

Стенд `bench/tantivy_m2v` (Tantivy 0.27 + model2vec-rs 0.5, сборка под процессор): индекс BM25 в памяти с языковыми анализаторами (стеммер и стоп-слова для английского, немецкого, арабского; биграммы символов для китайского), запрос — OR по токенам; векторы Model2Vec-RS по всем документам (точный поиск); слияние BM25/max + 0.6 × min-max(векторы) — как в остальных гибридах. Конкурент настроен в свою пользу: варианты анализатора выбирались по test. Время — от запуска процесса до готовности к поиску, на той же машине.

| nDCG@10 | SciFact | NFCorpus | FiQA | BEIR | en | zh | de | ar | hi | MLQA |
|---|---|---|---|---|---|---|---|---|---|---|
| Tantivy BM25 | 0.684 | 0.321 | 0.250 | 0.418 | 0.728 | 0.693 | 0.592 | 0.560 | 0.563 | 0.627 |
| Tantivy + Model2Vec-RS potion-base-8M | 0.706 | 0.335 | 0.267 | 0.436 | 0.741 | 0.691 | 0.595 | 0.559 | 0.563 | 0.630 |
| Tantivy + Model2Vec-RS potion-multilingual-128M | 0.702 | 0.329 | 0.260 | 0.430 | 0.735 | 0.706 | 0.611 | 0.580 | 0.586 | 0.644 |
| synrag BM25 | 0.695 | 0.317 | 0.261 | 0.424 | 0.752 | 0.689 | 0.604 | 0.569 | 0.598 | 0.642 |
| **synrag быстрый** (BM25 + Synapse) | 0.706 | 0.338 | 0.279 | **0.441** | 0.762 | 0.696 | 0.611 | 0.578 | 0.607 | **0.651** |
| **synrag быстрый + переранжировщик** | 0.714 | 0.342 | 0.296 | **0.451** | 0.777 | 0.704 | 0.620 | 0.580 | 0.617 | **0.660** |

| FiQA, 57.6 тыс. документов | Готов к поиску | Кодирование | Память | Запрос p50 |
|---|---|---|---|---|
| Tantivy + Model2Vec-RS potion-base-8M | 0.44 с | 750 тыс. док/с | 370 МБ | 3.7 мс |
| Tantivy + Model2Vec-RS potion-multilingual-128M | 2.8 с | 56 тыс. док/с | 1.85 ГБ | 3.8 мс |
| synrag быстрый | 1.6 с | 55 тыс. док/с | 443 МБ | 0.55 мс |
| synrag быстрый + переранжировщик | 1.4–1.6 с | 55 тыс. док/с | 443 МБ | 2.9 мс |

- Английский: synrag лучше (BEIR +0.005…+0.011, MLQA en +0.021…+0.027; с переранжировщиком BEIR +0.015…+0.021).
- С переранжировщиком synrag лучше по среднему пяти языков MLQA (+0.016): английский +0.036, хинди +0.031, немецкий +0.009; китайский −0.002, арабский вровень (0.580). Задержка запроса тогда ~3 мс — на уровне стенда.
- Пять языков: synrag лучше на 0.007 (английский +0.027, хинди +0.021, немецкий 0.000; китайский −0.010, арабский −0.002). В версии 1.2 synrag отставал на 0.010 из-за арабского (−0.063): у Tantivy был арабский стеммер, у synrag нет. В 1.3 в анализатор добавлены лёгкие стеммеры для арабского и хинди.
- Скорость старта и кодирования — у конкурента не хуже; задержка запроса у synrag ниже (в стенде векторы ищутся полным перебором в одном потоке; с ANN-индексом конкурент был бы быстрее).
- Чего у стека нет из коробки: парсеры PDF/Office, сжатый контекст с цитатами, HTTP-сервер для агента, один файл поставки — это нужно дописать (интеграция, не исследование).

### 7б. Языки СНГ и Ближнего Востока: 12 языков MQA

Вопросы и ответы пользователей (MQA) на одном языке; отложенные пары (5 %, при обучении Synapse не использовались), документы — их ответы плюс до 20 тыс. посторонних ответов того же языка; вопросы поделены пополам на dev (подбор настроек) и test (цифры ниже). Данные шумные (много школьных «помогите решить»), поэтому абсолютные значения ниже, чем на MLQA; сравнение — на одних и тех же данных. Synapse учился на других парах из того же источника, поэтому для него эти данные «родные»; столбец BM25 от этого не зависит. Tantivy — со стеммерами для турецкого, русского, английского, немецкого, арабского; для остальных без стеммера (у Tantivy их нет). Elasticsearch и модели эмбеддингов (версия 1.7) — как в разделах 6–7: `english` analyzer, модели на GPU, гибрид с весом 0.6.

| test, nDCG@10 | tr | kk | uz | az | ky | ru | en | de | ar | fa | ur | hi | Среднее |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Вопросов | 1463 | 1078 | 206 | 511 | 94 | 1473 | 1476 | 1512 | 1576 | 1541 | 812 | 1454 | |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Tantivy BM25 | 0.566 | 0.254 | 0.436 | 0.436 | 0.055 | 0.415 | 0.557 | 0.467 | 0.189 | 0.421 | 0.663 | 0.549 | 0.417 |
| Tantivy + Model2Vec-RS multilingual | 0.569 | 0.252 | 0.452 | 0.459 | 0.053 | 0.433 | 0.565 | 0.471 | 0.194 | 0.430 | 0.695 | 0.547 | 0.427 |
| **synrag BM25** | 0.577 | 0.286 | 0.501 | 0.511 | 0.071 | 0.435 | 0.573 | 0.462 | 0.201 | 0.435 | 0.716 | 0.541 | **0.442** |
| **synrag быстрый** (BM25 + Synapse) | 0.580 | 0.304 | 0.509 | 0.482 | 0.061 | 0.486 | 0.604 | 0.471 | 0.216 | 0.457 | 0.780 | 0.566 | **0.460** |
| **synrag быстрый + переранжировщик** | 0.587 | 0.324 | 0.515 | 0.524 | 0.061 | 0.527 | 0.624 | 0.485 | 0.217 | 0.477 | 0.801 | 0.589 | **0.478** |
| Elasticsearch BM25 | 0.551 | 0.271 | 0.438 | 0.461 | 0.062 | 0.373 | 0.560 | 0.452 | 0.160 | 0.427 | 0.692 | 0.544 | 0.416 |
| multilingual-e5-small (GPU) | 0.585 | 0.169 | 0.333 | 0.365 | 0.063 | 0.530 | 0.638 | 0.464 | 0.160 | 0.472 | 0.596 | 0.534 | 0.409 |
| **EmbeddingGemma 2 (GPU)** | 0.608 | 0.290 | 0.461 | 0.489 | 0.055 | 0.573 | 0.664 | 0.474 | 0.242 | 0.531 | 0.777 | 0.665 | **0.486** |
| Elasticsearch + e5-small | 0.566 | 0.290 | 0.452 | 0.479 | 0.071 | 0.427 | 0.598 | 0.484 | 0.172 | 0.459 | 0.715 | 0.559 | 0.439 |
| Elasticsearch + EmbeddingGemma | 0.568 | 0.292 | 0.447 | 0.494 | 0.060 | 0.435 | 0.608 | 0.482 | 0.186 | 0.479 | 0.711 | 0.579 | 0.445 |
| all-MiniLM-L6-v2 (GPU) | 0.238 | 0.053 | 0.270 | 0.152 | 0.004 | 0.112 | 0.620 | 0.308 | 0.022 | 0.022 | 0.441 | 0.194 | 0.203 |
| bge-small-en-v1.5 (GPU) | 0.439 | 0.096 | 0.390 | 0.292 | 0.004 | 0.227 | 0.644 | 0.335 | 0.030 | 0.060 | 0.528 | 0.165 | 0.267 |

- Один BM25 synrag лучше всей связки Tantivy + Model2Vec (0.442 против 0.427): анализатор synrag сильнее на казахском (+0.032), узбекском (+0.065), азербайджанском (+0.075), урду (+0.053).
- С Synapse — 0.460 (+0.033 к Tantivy + Model2Vec). Хуже Tantivy только немецкий BM25 (−0.005) и хинди BM25 (−0.008).
- С переранжировщиком — 0.478 (+0.051 к Tantivy + Model2Vec): русский +0.041, азербайджанский +0.042, казахский +0.020, остальные 0.000…+0.023. Переранжировщик учился на других (не отложенных) парах MQA, поэтому этот набор для него «родной», как и для Synapse; на BEIR и MLQA обучающих данных из тех же источников нет.
- **Против EmbeddingGemma на GPU** (0.486): synrag в среднем ниже на 0.008. synrag лучше на казахском (+0.034), узбекском (+0.054), азербайджанском (+0.035), урду (+0.024), немецком (+0.011); Gemma лучше на хинди (−0.076), персидском (−0.054), русском (−0.046), английском (−0.040), арабском (−0.025), турецком (−0.021). Тюркские языки с малыми ресурсами — сильная сторона synrag (анализатор + Synapse), большие языки — сторона Gemma.
- Гибрид Elasticsearch + EmbeddingGemma (0.445) хуже одной Gemma: английский analyzer Elasticsearch на этих языках слаб, а вес 0.6 не подбирался. Гибриды в этом отчёте — типовая настройка, а не лучшая возможная.
- Английские модели (MiniLM, bge-small) вне английского почти не работают (0.20–0.27).
- Киргизский набор мал (94 вопроса) и очень шумный — для выводов не годится.

**Вопрос на одном языке, документы на другом** (вопрос по-русски / по-китайски, 21 тыс. английских абзацев): nDCG@10 0.20 / 0.16 против 0.86 для английского вопроса — работает, но слабо; это главная зона роста Synapse.

### 7в. Независимые наборы: ещё 4 набора BEIR и Belebele (14 языков)

Проверка на данных, которые не участвовали ни в обучении, ни в настройке: настройки synrag те же, что выбраны на dev раньше (быстрый режим, вес Synapse 0.6; переранжировщик 0.4 на 30 кандидатах), ничего не подбиралось. Тексты FLORES-200, на которых построен Belebele, и наборы BEIR ниже в обучающих данных Synapse и переранжировщика не встречаются. Belebele (Meta, CC BY-SA 4.0) в постановке поиска, как BelebeleRetrieval в MTEB: вопрос на понимание текста → нужный отрывок среди 488 отрывков того же языка, 900 вопросов на язык. Elasticsearch: на BEIR — `english` analyzer; на Belebele — встроенный языковой analyzer, где он есть (en, ru, de, ar, fa, hi, tr, zh — `cjk`), иначе `standard` (для uk, kk, uz, az, ky, tg встроенных нет). Модели — GPU, длина 256 токенов; гибриды с Elasticsearch — вес 0.6, как в разделах 6–7б.

**Английский: SciDocs, ArguAna, TREC-COVID, Quora** (test, nDCG@10)

| | SciDocs | ArguAna | TREC-COVID | Quora | Среднее |
|---|---|---|---|---|---|
| Документов / вопросов | 25.7 тыс. / 1000 | 8.7 тыс. / 1406 | 171 тыс. / 50 | 523 тыс. / 10 000 | |
| **synrag BM25** | 0.156 | 0.367 | 0.637 | 0.787 | **0.487** |
| **synrag быстрый** | 0.166 | 0.344 | 0.703 | 0.834 | **0.512** |
| **synrag быстрый + переранжировщик** | 0.165 | 0.347 | 0.713 | 0.846 | **0.518** |
| Elasticsearch BM25 | 0.165 | 0.352 | 0.688 | 0.808 | 0.503 |
| Tantivy BM25 | 0.156 | 0.356 | 0.604 | 0.808 | 0.481 |
| Tantivy + Model2Vec-RS | 0.157 | 0.372 | 0.645 | 0.835 | 0.502 |
| multilingual-e5-small (GPU) | 0.144 | 0.300 | 0.729 | 0.882 | 0.514 |
| EmbeddingGemma 2 (GPU) | 0.179 | 0.461 | 0.586 | 0.872 | 0.524 |

| | | | | | |
|---|---|---|---|---|---|
| all-MiniLM-L6-v2 (GPU) | 0.217 | 0.356 | 0.470 | 0.875 | 0.480 |
| bge-small-en-v1.5 (GPU) | 0.196 | 0.399 | 0.745 | 0.886 | 0.557 |
| Elasticsearch + e5-small | 0.175 | 0.371 | 0.747 | 0.855 | 0.537 |
| Elasticsearch + EmbeddingGemma | 0.179 | 0.426 | 0.777 | 0.861 | 0.561 |
| Elasticsearch + MiniLM | 0.188 | 0.383 | 0.750 | 0.860 | 0.545 |
| Elasticsearch + bge-small | 0.184 | 0.401 | 0.750 | 0.860 | 0.549 |
| Готов к поиску: synrag быстрый, CPU | 0.9 с | 0.6 с | 6.6 с | 4.8 с | |
| Кодирование на GPU: e5-small / bge-small / EmbeddingGemma | 39 / 46 / 165 с | 16 / 12 / 50 с | 271 / 215 / 1030 с | 92 / 161 / 676 с | |

- Здесь synrag слабее, чем на SciFact, NFCorpus и FiQA (раздел 6). Быстрый + переранжировщик (0.518) выше Elasticsearch (0.503), Tantivy + Model2Vec (0.502) и MiniLM (0.480), на уровне e5-small (0.514) и одной EmbeddingGemma (0.524), но **ниже bge-small (0.557) и всех гибридов Elasticsearch + модель на GPU (0.537–0.561)**.
- **ArguAna** (запрос — целый аргумент, найти контраргумент): Synapse здесь вредит, 0.367 → 0.344; лучшая — EmbeddingGemma (0.461). Synapse учился на парах «вопрос → ответ», такой задачи в обучении не было.
- **SciDocs** (по названию статьи найти статьи, которые её цитируют): у всех 0.14–0.22, лучше всех английские модели MiniLM и bge-small; synrag на уровне Elasticsearch.
- **TREC-COVID** (50 вопросов, 171 тыс. статей): synrag 0.713 — выше Elasticsearch (0.688) и одной EmbeddingGemma (0.586), ниже e5-small и bge-small (0.73–0.75) и гибридов (0.75–0.78). Значение одной EmbeddingGemma ниже публикуемых — вероятно, из-за обрезки статей до 256 токенов в нашей настройке; 50 вопросов — большой разброс.
- **Quora** (найти вопрос-дубликат): synrag 0.846 — выше Elasticsearch (0.808), ниже моделей (0.872–0.886).
- Скорость: база Quora (523 тыс. вопросов) готова к поиску на CPU за 4.8 с; кодирование моделями на GPU — 1.5–11 мин.

**14 языков: Belebele** (test, nDCG@10; 900 вопросов и 488 отрывков на язык)

| | en | ru | uk | kk | uz | az | ky | tg | tr | de | ar | fa | hi | zh | Среднее |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **synrag BM25** | 0.926 | 0.914 | 0.883 | 0.788 | 0.827 | 0.788 | 0.817 | 0.845 | 0.838 | 0.864 | 0.852 | 0.809 | 0.764 | 0.911 | **0.845** |
| **synrag быстрый** | 0.951 | 0.925 | 0.902 | 0.808 | 0.838 | 0.801 | 0.831 | 0.853 | 0.868 | 0.884 | 0.874 | 0.851 | 0.783 | 0.923 | **0.864** |
| **synrag быстрый + переранжировщик** | 0.960 | 0.931 | 0.908 | 0.834 | 0.847 | 0.818 | 0.838 | 0.856 | 0.880 | 0.883 | 0.880 | 0.879 | 0.792 | 0.931 | **0.874** |
| Elasticsearch BM25 | 0.917 | 0.914 | 0.698 | 0.653 | 0.719 | 0.638 | 0.666 | 0.761 | 0.852 | 0.883 | 0.858 | 0.843 | 0.774 | 0.894 | 0.791 |
| Tantivy BM25 | 0.916 | 0.914 | 0.699 | 0.652 | 0.702 | 0.633 | 0.668 | 0.761 | 0.836 | 0.886 | 0.848 | 0.813 | 0.737 | 0.912 | 0.784 |
| Tantivy + Model2Vec-RS | 0.928 | 0.923 | 0.754 | 0.705 | 0.746 | 0.696 | 0.700 | 0.774 | 0.865 | 0.907 | 0.869 | 0.844 | 0.777 | 0.922 | 0.815 |
| multilingual-e5-small (GPU) | 0.948 | 0.921 | 0.912 | 0.752 | 0.705 | 0.769 | 0.723 | 0.479 | 0.901 | 0.924 | 0.882 | 0.916 | 0.862 | 0.927 | 0.830 |
| EmbeddingGemma 2 (GPU) | 0.946 | 0.934 | 0.925 | 0.734 | 0.721 | 0.813 | 0.732 | 0.619 | 0.914 | 0.933 | 0.901 | 0.915 | 0.903 | 0.939 | 0.852 |
| Elasticsearch + e5-small | 0.948 | 0.941 | 0.782 | 0.731 | 0.781 | 0.721 | 0.735 | 0.793 | 0.900 | 0.919 | 0.902 | 0.892 | 0.837 | 0.924 | 0.843 |
| Elasticsearch + EmbeddingGemma | 0.954 | 0.949 | 0.800 | 0.734 | 0.791 | 0.743 | 0.746 | 0.801 | 0.907 | 0.924 | 0.908 | 0.901 | 0.848 | 0.934 | 0.853 |

- **В среднем по 14 языкам synrag впереди всех**: быстрый + переранжировщик 0.874, быстрый 0.864 против EmbeddingGemma на GPU 0.852, гибрида Elasticsearch + EmbeddingGemma 0.853, Elasticsearch + e5 0.843, Tantivy + Model2Vec 0.815, Elasticsearch 0.791. На английском synrag тоже первый (0.960; MiniLM 0.916, bge-small 0.912, их гибриды 0.947–0.952).
- **Перевес даёт группа языков с малыми ресурсами**: против EmbeddingGemma таджикский +0.236, узбекский +0.126, киргизский +0.106, казахский +0.100, азербайджанский +0.005, английский +0.015. **EmbeddingGemma лучше на 8 языках из 14**: хинди −0.111, немецкий −0.050, персидский −0.036, турецкий −0.034, арабский −0.021, украинский −0.017, китайский −0.008, русский −0.003. Без казахского, узбекского, киргизского и таджикского: EmbeddingGemma 0.912, synrag 0.886.
- Вопросы Belebele написаны по тексту отрывка и делят с ним много слов — это выгодно BM25: один BM25 synrag (0.845) почти равен EmbeddingGemma (0.852). Главная причина отрыва — анализатор synrag: там, где у Elasticsearch и Tantivy нет стеммера, BM25 synrag выше на 0.08–0.18 (казахский 0.788 против 0.653). Synapse и переранжировщик добавляют к BM25 synrag ещё +0.029.

- Разрывы на казахском, узбекском, киргизском и таджикском (0.10–0.24 при 900 вопросах) много больше случайного разброса; разрывы меньше ~0.02 (русский, китайский, азербайджанский) значимыми считать нельзя (бутстреп не проводился).
- База одного языка (488 отрывков) готова к поиску на CPU за 0.2–0.4 с.
- Гибриды Elasticsearch + модель на украинском, казахском и азербайджанском хуже одной модели: у Elasticsearch нет встроенного анализатора для этих языков, а вес 0.6 не подбирался.

---

## 8. Задержка, пропускная способность, память, размер

| | synrag | Elasticsearch | Модели эмбеддингов |
|---|---|---|---|
| Задержка запроса p50 | BM25 0.03–0.36 мс; полный 0.15–0.9 мс; с переранжировщиком ~2.9 мс (FiQA; в процессе) | 2.2–4.2 мс (HTTP) | 4–28 мс (кодирование запроса + поиск, Python) |
| Пропускная способность | 7–190 тыс. запросов/с (12 потоков); с переранжировщиком ~0.5 тыс./с | 2.2–4.1 тыс./с (12 клиентов HTTP) | — |
| Память при работе | 61–475 МБ | 4.9 ГБ | 1.6–2.8 ГБ ОЗУ + 0.7–5.6 ГБ GPU |
| Индекс, FiQA | 60 МБ (BM25 с текстом) / 90 МБ (полный) | 43.5 МБ | векторы 88–177 МБ (f32) + текст отдельно |
| Поставка | 1.75 МБ + модель 86 МБ | образ 1.27 ГБ + JVM | PyTorch/CUDA (несколько ГБ) + модель 90 МБ – 1.5 ГБ |

Строки «в процессе» выше — synrag вызывается как библиотека; ниже — оба сервера по HTTP, одним клиентом.

**HTTP против HTTP** (FiQA test, 648 вопросов; один и тот же клиент на Python — `bench/http_bench.py`; задержка — один клиент, вопрос за вопросом; пропускная способность — 12 клиентов; свободный сервер; nDCG@10 посчитан по ответам самих серверов):

| Сервер | nDCG@10 | p50 / p95 / p99 | 12 клиентов | Память сервера |
|---|---|---|---|---|
| synrag BM25 ( `serve --bm25` ) | 0.256 | **0.47 / 0.73 / 0.86 мс** | 4 740 запр/с | 116 МБ |
| **synrag быстрый** (BM25 + Synapse) | **0.279** | **1.35 / 1.80 / 2.05 мс** | **4 134 запр/с** | **171 МБ** |
| synrag быстрый + переранжировщик | **0.296** | 3.17 / 3.84 / 4.10 мс | 475 запр/с | 296 МБ |
| Elasticsearch 8.15, новое соединение на запрос | 0.254 | 2.36 / 4.03 / 4.75 мс | 2 481 запр/с | 4.7 ГБ |
| Elasticsearch 8.15, постоянное соединение | 0.254 | 1.96 / 3.72 / 4.34 мс | 3 197 запр/с | 4.7 ГБ |
| **synrag быстрый, сервер версии 1.9**, новое соединение | **0.279** | **1.01 / 1.43 / 1.63 мс** | 4 205 запр/с | 315 МБ |
| **synrag быстрый, сервер версии 1.9**, постоянное соединение | **0.279** | **0.76 / 1.07 / 1.23 мс** | **6 325 запр/с** | 315 МБ |

- Против лучшего варианта Elasticsearch (постоянное соединение) synrag быстрый: качество +0.025, задержка p50 в 1.5 раза и p99 в 2.1 раза ниже, пропускная способность в 1.3 раза выше, памяти в ~27 раз меньше. BM25 против BM25 — в 4 раза быстрее по p50. Сервер synrag закрывает соединение после каждого ответа (постоянных соединений пока нет) — и всё равно быстрее.
- Пропускная способность лёгких режимов упирается в клиент на Python (один процесс), а не в сервер: реальный потолок synrag выше (внутри процесса — 11.7 тыс. запр/с, раздел 8).
- С переранжировщиком качество выше Elasticsearch на 0.042, задержка p50 3.2 мс — на уровне p95 Elasticsearch, но пропускная способность в ~6.7 раза ниже (~11.5 мс CPU на запрос).
- Память сервера — уже построенный индекс, загруженный с диска (при построении в одном процессе пик выше: 443 МБ).

Переранжировщик (FiQA, свободный сервер): в один поток запрос занимает 12.1 мс против 0.58 мс без него — это и есть процессорное время на запрос (~11.5 мс на 30 документов × 256 токенов через Synapse и проекции). На 12 потоках кандидаты считаются параллельно: p50 3.0 мс, 478 запросов/с против 11.7 тыс. без переранжировщика (в ~24 раза меньше). (В первой редакции версии 1.7 стояла оценка «~30 мс CPU» — она была неверной и заменена замером.)

---

## 8a. Обновление базы без перестройки

Документы добавляются, заменяются (тот же `source` — новая версия) и удаляются на лету: CLI `synrag add` / `synrag delete`, HTTP `POST /add`, `/delete`, `/save`, библиотека `Engine::upsert` / `delete`. Как устроено: разбор и векторизация новых документов (Synapse) идут без блокировки, параллельно с поиском; под блокировкой только слияние — новые

постинги дописываются в небольшую дельту, удаление помечает чанки и точно вычитает их статистику (df, длины, idf остаются как при полной перестройке). Дельта вливается в основные массивы, когда достигает 1/8 их размера; удалённые чанки физически убираются при доле > 25 %.

FiQA, 57 638 документов, BM25 + Synapse, тот же бокс (`examples/update_bench.rs`):

| Операция | Подготовка (без блокировки) | Под блокировкой | Итого |
|---|---|---|---|
| добавить 1 документ | 0.21 мс | 0.01 мс | 0.2 мс |
| добавить 100 / 1000 документов | 2.8 / 21 мс | 0.6 / 5.4 мс | 3.4 / 26 мс |
| заменить 1 / 100 / 1000 документов | — | — | 0.2 / 4.6 / 30 мс |
| удалить 1 / 100 / 1000 документов | — | — | < 0.05 / 0.1 / 0.9 мс |
| уплотнение после 2 099 удалений | — | 25 мс | 25 мс |
| полная перестройка (для сравнения) | — | — | 1 180 мс |

| Поиск, 4 потока | p50 | p99 |
|---|---|---|
| без обновлений | 0.56 мс | 0.99 мс |
| одновременно ~1 100 замен документов в секунду | 0.87 мс | 1.49 мс |

- Качество: база из 53 438 документов + 4 200 добавленных по одному и пачками даёт nDCG@10 0.2788 — ровно как полная перестройка; тесты проверяют совпадение оценок и статистик терм в терм после добавлений, удалений, замен, слияния, уплотнения и сохранения.
- Не обновляется на лету модель корпуса (`--learn`): новые документы кодируются старыми весами; при сильном изменении корпуса её стоит переобучить перестройкой. Быстрый режим (BM25 + Synapse) этого не требует.
- **Журнал изменений** (версия 1.6): каждое обновление сначала дописывается в `<индекс>.wal` с контрольной суммой и сбрасывается на диск, затем применяется; при запуске synrag загружает снимок и проигрывает журнал; `/save` пишет снимок и обнуляет журнал; оборванная при сбое последняя запись отбрасывается. Проверено `kill -9` сервера без сохранения: все подтверждённые обновления восстановлены.

---

## 8б. Масштаб: MS MARCO, 8.8 млн отрывков

MS MARCO passage (BEIR): 8 841 823 отрывка (3.0 ГБ текста, в среднем ~340 байт — это короткие отрывки, а не полноразмерные документы; длинные документы synrag режет на чанки, и масштаб считается в чанках), 6 980 вопросов dev. Тот же сервер: 6 ядер, 31 ГБ ОЗУ, GPU не используется. Время — от запуска процесса до готовности к поиску; задержка — один поток, вопрос за вопросом.

| Режим | Готов к поиску | nDCG@10 | Recall@100 | Запрос p50 / p99 | 12 потоков | Пик памяти |
|---|---|---|---|---|---|---|
| synrag BM25 | **22 с** | 0.244 | 0.674 | 7.1 / 28 мс | 450 запр/с | 15.2 ГБ |
| synrag BM25 + Synapse, IVF 16 | 143 с | 0.276 | 0.725 | 8.7 / 30 мс | 401 запр/с | 16.2 ГБ |
| **synrag BM25 + Synapse, IVF 32** (по умолчанию) | 142 с | **0.277** | 0.730 | 9.9 / 31 мс | 372 запр/с | 16.2 ГБ |
| synrag BM25 + Synapse, IVF 64 | 141 с | 0.277 | 0.734 | 12.0 / 33 мс | 326 запр/с | 16.1 ГБ |
| synrag BM25 + статический Synapse, IVF 32 | **59 с** | 0.251 | 0.686 | 10.3 / 32 мс | 359 запр/с | 16.2 ГБ |

- Из 142 с старта: чтение 5 с, индекс BM25 16 с, Synapse 120 с (74 тыс. док/с), индекс IVF 2.7 с.
- BM25 synrag (0.244) выше опубликованного BM25 в BEIR (0.228). Synapse добавляет +0.033.
- Модели, обученные на самом MS MARCO (по статье BEIR: TAS-B 0.408, ANCE 0.388), здесь сильнее — для них это родной домен; synrag на этих данных не обучался.
- **IVF** (приближённый векторный поиск, версия 1.6): двухуровневая кластеризация векторов, запрос смотрит 32 из 4 422 кластеров (~0.7 % документов). На 1 млн отрывков MS MARCO качество равно полному перебору (nDCG@10 0.5794 против 0.5796), а запрос в 2.2 раза быстрее (5.3 против 11.7 мс); на 8.8 млн полный перебор дал бы ~100 мс на запрос.
- Память: потоковое чтение, освобождение документов при нарезке, компактные номера термов, поблочное квантование векторов и возврат памяти системе снизили пик с 18.5 до 15.2 ГБ (BM25); после построения процесс держит 11 ГБ. Это первый замер такого масштаба: распределённый режим и 100 млн не проверялись.

---

## 8в. Большие данные: что готово и чего нет

Вопросы, которые задаст платформа данных (Databricks, AWS и подобные), и ответы по замерам.

| Вопрос | Ответ |
|---|---|
| Сколько текста на один сервер | Проверено 8.84 млн коротких отрывков MS MARCO (~340 байт каждый, 3.0 ГБ текста) на 6 ядрах и 31 ГБ: BM25 готов за 22 с, с Synapse — за 2.4 мин; пик памяти 15–16 ГБ, после построения 11 ГБ (~1.3 КБ на отрывок вместе с текстом). Для полноразмерных документов считать надо чанки (~100–200 слов) |
| Скорость запросов на таком объёме | p50 7–10 мс, 372–450 запросов/с на 12 потоках (раздел 8б) |
| 100 млн чанков | Только оценка: кодирование Synapse ~30 мин на сервер; памяти ~125 ГБ после построения, пик ~180 ГБ — нужен шардинг |
| Распределённый режим, шардинг, реплики | **Есть (версия 1.9)**: координатор, шарды по хэшу источника, реплики с переключением при сбое; FiQA на 2 шардах — nDCG 0.2777 против 0.2788 на одном сервере, p50 1.3 мс; общая статистика слов (idf) по всем шардам (раздел 8д). Нет автоматического перешардирования |
| Spark, Delta Lake, Parquet, S3 | **Коннекторы (версия 1.9)**: S3 и совместимые хранилища, Confluence, SharePoint / OneDrive, Google Drive, любые SQL-базы, папки — с переносом прав; загрузка файлов по HTTP ( `/upload` ). Коннекторов к Spark / Delta Lake / Parquet нет: таблицу можно отдать через SQL-коннектор |
| Табличные данные (миллиарды строк, SQL) | Не задача synrag: он ищет по тексту. Строки таблиц можно индексировать как текст, аналитику — нет |
| Обновления и надёжность | Добавление документа 0.2 мс без перестройки, журнал изменений с восстановлением после сбоя (проверено `kill -9` ); один процесс-писатель; снимок пишется целиком |
| Права на документы, аудит, шифрование, SSO | **Есть (версия 1.9)**: права на каждый документ (+0.24 мс на запрос), журнал аудита с цепочкой хэшей, шифрование хранения, SSO-токены OIDC и ключи API с ролями. TLS — на ingress / прокси; SAML — через мост OIDC (Keycloak) |
| Стоимость кодирования | 1 млн документов размера FiQA: ~$0.003 на AWS c7a.2xlarge против $0.12 (e5-small на L4), $1.08 (EmbeddingGemma на L4), $3.8 (API OpenAI / Titan) — раздел 8г. CPU вместо GPU и платных API: 51.6 тыс. док/с на 6 ядрах — в 410–780 раз больше трансформерных моделей на том же CPU (ONNX int8) и в 23–29 раз больше их fp16 на GPU |
| Качество против эмбеддингов платформ | Открытые модели — раздел 1а: на английском synrag уступает 0.02–0.05 nDCG@10, на 5 языках на уровне EmbeddingGemma, на 12 языках MQA уступает ей 0.008 в среднем. Проприетарные модели (Titan, OpenAI и др.) не мерились |

Итог (версия 1.9): synrag работает и на одном сервере, и кластером из шардов с репликами, с правами доступа, аудитом и SSO. Проверены 2 шарда на одной машине и 8.8 млн отрывков на одном сервере; кластеры из десятков узлов и сотни миллионов документов не проверялись. Рядом с платформами данных (Databricks, AWS) synrag — дешёвый поиск и кодирование на CPU; аналитику по таблицам он не заменяет.

## 8г. Стоимость индексации в долларах

Сколько стоит превратить 1 млн документов размера FiQA (~760 байт, ~190 токенов) в готовую к поиску базу. Время — из наших замеров (разделы 4–5), умноженное на цену облачного сервера. Цены — AWS us-east-1 по требованию, по сторонним агрегаторам цен на ноябрь 2026 (перед бюджетированием сверить с официальными страницами): CPU c7a.2xlarge (8 ядер AMD Zen 4) — $0.41/ч; GPU g6.xlarge (NVIDIA L4) — $0.80/ч; API эмбеддингов OpenAI text-embedding-3-small и Amazon Titan Text Embeddings V2 — $0.02 за 1 млн токенов, Cohere Embed — $0.08–0.12.

| Способ | Сервер | Скорость | Время на 1 млн | $ за 1 млн документов | $ за 1 млрд |
|---|---|---|---|---|---|
| synrag BM25 (вся база) | c7a.2xlarge | — | 8 с | **$0.001** | ~$1 |
| synrag турбо (вся база) | c7a.2xlarge | 235 тыс. док/с | 13 с | **$0.0015** | ~$1.5 |
| **synrag быстрый: BM25 + Synapse (вся база)** | c7a.2xlarge | 51.6 тыс. док/с | 28 с | **$0.003** | **~$3** |
| all-MiniLM-L6-v2, fp16 (только векторы) | g6.xlarge | 2 252 док/с | 7.4 мин | $0.10 | ~$100 |
| multilingual-e5-small, fp16 (только векторы) | g6.xlarge | 1 818 док/с | 9.2 мин | $0.12 | ~$120 |
| EmbeddingGemma 2 (только векторы) | g6.xlarge | 207 док/с | 1.3 ч | $1.08 | ~$1 080 |
| multilingual-e5-small, ONNX int8 (только векторы) | c7a.2xlarge | 71 док/с | 3.9 ч | $1.61 | ~$1 610 |
| EmbeddingGemma 2 на CPU (только векторы) | c7a.2xlarge | 9 док/с | 31 ч | $12.7 | ~$12 700 |

| | | | | | |
|---|---|---|---|---|---|
| API OpenAI text-embedding-3-small / Amazon Titan V2 | — | — | — | $3.8 | ~$3 800 |
| API Cohere Embed | — | — | — | $15–23 | ~$15–23 тыс. |

- synrag дешевле: в ~35–40 раз, чем e5-small или MiniLM на GPU; в ~350 раз, чем EmbeddingGemma на GPU; в ~500 раз, чем e5-small на том же CPU; в ~1 200 раз, чем API OpenAI или Titan. При этом у synrag в цену входит вся база (BM25 + векторы), а у конкурентов — только векторы, без индекса Elasticsearch или векторной базы.
- В абсолютных деньгах на 1 млн документов разница мала (центы против долларов); она становится заметной на сотнях миллионов документов и при частой переиндексации (смена модели, новые версии документов).
- Допущения: скорость c7a.2xlarge (8 физических ядер Zen 4) принята равной нашему 6-ядерному Ryzen 5 7500F той же архитектуры — скорее занижает synrag; L4 принята равной RTX 5060 Ti — реальная скорость может отличаться в ~1.5 раза в любую сторону; ~4 байта на токен (английский). Не учтены хранение, обслуживание запросов и кодирование запросов; качество систем разное (раздел 1а).

---

## 8д. Enterprise: безопасность и эксплуатация (версия 1.9)

Всё — в том же исполняемом файле, без внешних зависимостей; руководство для служб эксплуатации и безопасности — `docs/ENTERPRISE.md`. Цена функций измерена на FiQA (57.6 тыс. документов, загружены через HTTP `/add`), тот же сервер, клиент на Python по HTTP с постоянным соединением, 648 вопросов.

| Режим сервера | nDCG@10 | p50 / p95 / p99 | 12 клиентов |
|---|---|---|---|
| без авторизации (индекс построен заранее) | 0.279 | 0.76 / 1.07 / 1.23 мс | 6 325 запр/с |
| ключ API, роль `read-all`, документы загружены через HTTP `/add` | **0.279** | 0.78 / 1.12 / 1.27 мс | 5 979 запр/с |
| + журнал аудита (запись в ОС на каждый запрос) | 0.279 | 0.83 / 1.16 / 1.32 мс | 6 186 запр/с |
| + журнал аудита с fsync каждой записи ( `--audit-sync on` ) | 0.279 | 1.80 / 2.16 / 2.34 мс | 994 запр/с |
| права на документы: пользователю видна половина базы (5 групп из 10) | —[1] | 1.05 / 1.40 / 1.55 мс | 5 449 запр/с |
| кластер: координатор + 2 шарда, общий idf | 0.277 | 1.49 / 1.84 / 1.99 мс | 3 066 запр/с |
| кластер: координатор + 2 шарда, idf шардов ( `"global_idf": false` ) | 0.278 | 1.33 / 1.62 / 1.80 мс | 3 366 запр/с |

[1] половина правильных ответов скрыта правами, поэтому nDCG с остальными строками не сравним. Строки с аудитом и правами мерились с нарезкой по 160 слов (качество не зависит от аудита); остальные — один фрагмент на документ, как во всех таблицах BEIR. С нарезкой по 160 слов FiQA даёт 0.268 и на одном сервере, и на 2 шардах (0.268).

**Исправление первой редакции версии 1.9.** Там стояли 0.276 для режима с ключом и 0.269 для кластера. Авторизация на качество не влияет: разница была в настройке нарезки (сервер по умолчанию режет документы по 160 слов, `beir` — один фрагмент на документ) и в ошибке скрипта замера — повтор одного документа в топ-10 засчитывался дважды. С той же нарезкой через HTTP с ключом получается ровно 0.2788. Сервер теперь возвращает один лучший фрагмент на документ ( `chunks=1` — все фрагменты). Потеря качества в кластере (было −0.007) возникала потому, что каждый шард нормирует оценки по своему пулу кандидатов; теперь шарды отдают границы нормировки, координатор сливает их — разница −0.001.

- **Авторизация** (ключ API или SSO-токен) стоит ~0.05 мс; проверенные токены кэшируются до истечения.
- **Права на документы** фильтруются внутри поиска до отбора лучших результатов: пользователь получает полную страницу из того, что ему можно видеть, а скрытые документы не попадают ни в выдачу, ни в контекст LLM. Цена — +0.24 мс: маска доступных фрагментов считается один раз на набор групп и версию индекса (без кэша было +1.3 мс).
- **Кластер**: документы распределяются по шардам по хэшу источника (28 639 и 28 961 документ), запись — на все реплики шарда, чтение — параллельно с переключением на другую реплику при сбое. Поиск в две фазы, как dfs_query_then_fetch в Elasticsearch: координатор собирает частоты слов запроса со всех шардов, шарды считают BM25 с общей статистикой, а координатор сливает оценки по объединённым границам нормировки. Результат на 2 шардах — 0.2767–0.2777 против 0.2788 на одном сервере. На FiQA общий idf почти ничего не меняет (раскладка по хэшу и так ровная), но на неравномерном корпусе он возвращает порядок одного сервера (тест: совпадение позиций 7 из 60 без него, 45+ из 60 с ним); стоит +0.15 мс.
- **Кластер на двух машинах** (ноутбук и сервер, сеть Wi-Fi с задержкой 3–210 мс): 2 шарда × 2 реплики, по реплике каждого шарда на каждой машине, координатор на ноутбуке, FiQA загружен через координатор, 12 клиентов на другой машине. Качество 0.2767 — то же, что на одной машине.

| Шаг (версия 1.11, постоянные соединения координатора) | p50 / p95 / p99 | запр/с | ошибки |
|---|---|---|---|
| обычная работа | 14.6 / 23.0 / 29.0 мс | 252 | 0 |

| | | | |
|---|---|---|---|
| падение реплики под нагрузкой | 16.1 / 23.8 / 32.1 мс | 219 | 0 |
| зависший узел (соединения принимает, не отвечает) | 15.9 / 26.3 / 31.9 мс | 314 | 0 |

В версии 1.10 координатор открывал новое соединение на каждый запрос: p99 101 / 227 / 406 мс в тех же шагах. Запись при упавшей реплике в режиме `write_ack: all` отклоняется (502), живая реплика её применяет; отставшая реплика восстанавливается копией снимка с соседней (29.9 МБ за 1.8 с) — вручную `/cluster/repair` или, в режиме `write_ack: available`, автоматически, как только она снова отвечает.

- **Канал между узлами под mTLS** (ghostunnel, TLS 1.3, клиентские сертификаты; шард слушает только 127.0.0.1): поиск ноутбук → координатор → TLS → сервер — nDCG 0.2788, p50 6.1 мс, p99 12.7 мс, 423 запр/с; сам TLS на одной машине — +0.07 мс. Подключение к шарду из сети, без TLS, без клиентского сертификата или с самоподписанным сертификатом — отклоняется. Без постоянных соединений тот же путь давал p99 1.3 с (рукопожатие TLS на каждый запрос) — отсюда пул соединений координатора. С узлами на разных машинах координатор не запустится, пока в `cluster.json` не указан тип канала (`mtls` или `plaintext`); Helm-чарт настраивает mTLS через Istio / Linkerd и NetworkPolicy.
- **Записи при сбоях и нескольких писателях** (версия 1.12): у каждой записи версия, уникальная для каждого координатора (номер узла в младших битах), — реплика применяет для каждого документа только более новую версию, удаление оставляет «надгробие»; реплики сходятся к одному состоянию при любом порядке доставки и повторах. Для строгой согласованности — условная запись `if_version` («записать, только если текущая версия равна V»; иначе 409, запрос не применяется целиком). Условную запись решает одна реплика — первичная реплика шарда: она проверяет условие и назначает версию, остальные получают уже решённую запись (версия 1.13). Проверка: 400 раундов (по 200 в режимах `all` и `available`, прогон повторён 5 раз), в каждом два писателя одновременно через два разных координатора ожидают одну и ту же версию — всегда ровно один успех и один отказ 409, реплики одинаковы. Без связи с первичной репликой условная запись отказывает (503), поэтому при разделении сети двух успехов быть не может. **Первичная реплика, отставшая после разделения сети** (версия 1.14; сценарий рецензента выявил ошибку, тест её воспроизвёл): в режиме `available` обычная запись может быть подтверждена без первичной реплики; вернувшаяся первичная реплика принимала условную запись со старой версией по своей устаревшей копии и затирала подтверждённую запись. Теперь координатор сначала узнаёт версию документа у всех реплик шарда, и первичная реплика проверяет условие по самой новой: условная запись со старой версией получает 409 с новой версией, подтверждённая запись сохраняется, первичная реплика помечается отставшей и сливается с остальными; если какая-то реплика не отвечает — 503. В режиме `all` без первичной реплики запись не подтверждается, поэтому лишнего запроса нет. **Номер узла** (`node_id`) теперь обязателен (число или `"ordinal"` — номер из имени пода StatefulSet); координатор арендует его на всех шардах (30 с, продление каждые 3 с): второй координатор с тем же номером не запускается, координатор, потерявший аренду, не принимает записи (тест). Режимы подтверждения `all` / `available` с описанным контрактом сохранности (что переживает подтверждённая запись при падении процесса, потере питания и потере диска — `docs/ENTERPRISE.md`). **Разделение сети**: если обе стороны принимали записи, восстановление не копирует одну сторону поверх другой, а сливает их по версиям в обе стороны (документы с правами доступа и удаления) — проверено тестом: изменения разных документов с обеих сторон сохранились, конфликт решён в пользу более новой версии, более новое удаление победило, реплики одинаковы. Если обе стороны изменили **один и тот же** документ, остаётся только более позднее изменение («побеждает последний»), второе теряется — как в любой системе с таким правилом; где это недопустимо, нужна условная запись (при разделении сети она проходит только на стороне первичной реплики).
- **Внешние якоря**: контрольные точки журнала аудита с HMAC во внешнее хранилище (полностью переписанный журнал обнаруживается); якорь состояния каждой коллекции вне диска с данными — сервер отказывается стартовать на откаченных данных (восстановленная копия, откат тома), пока откат не подтверждён явно; ротация ключа шифрования `synrag rekey` (повторный запуск после прерывания безопасен). Якорь по умолчанию сбрасывается на диск при каждой записи.
- **Скорость записи со всеми гарантиями** (версия 1.13; ноутбук с NVMe, только BM25, HTTP): по одному документу от 12 клиентов — p50 3.0–3.3 мс, p99 7–10 мс, ~3.3–3.7 тыс. док/с во всех режимах (запись в коллекцию идёт по одной); пачками по 100 — 69 тыс. док/с по умолчанию (fsync журнала), 49 тыс. со всеми гарантиями (fsync, аудит, шифрование, якорь с fsync), 43 тыс. в кластере из 2 реплик с подтверждением `all`. На серверных дисках числа будут другими. **Повтор (версия 1.14)**: 5 полных прогонов и 3 пары «старая сборка / новая» вперемешку, ноутбук от батареи в режиме энергосбережения (поэтому абсолютные числа ниже). Медианы: по одному документу 2.2–2.6 тыс. док/с во всех режимах одного сервера — разница между режимами меньше разброса между прогонами (1.8–2.9 тыс.), так что прежнее «со всеми гарантиями чуть быстрее» — шум, а не эффект; пачками — 37–41 тыс. по умолчанию, 32–36 тыс. со всеми гарантиями (−10…−15 %; в однократном замере от сети было −29 %), 27–29 тыс. в кластере. Новая сборка против старой: медианы совпадают в пределах ±3 % (например, все гарантии пачками 35.6 против 35.7 тыс.) — исправления 1.14 запись не замедлили.
- **Обновление**: индекс и незакрытый журнал версии 1.8 открыты версией 1.13, журнал проигран, сохранено в новом формате; старая версия новый формат не читает (откат — только из резервной копии). Порядок обновления кластера, восстановление резервной копии при внешнем якоре — `docs/ENTERPRISE.md`. Адрес клиента в аудите: `X-`

`Forwarded-For` учитывается только от указанных адресов прокси и читается справа (клиент не может подделать свой адрес).

- **Утечки по времени выполнения**: статистический тест в духе dudect (`examples/ct_timing.rs`, 1 млн замеров на проверку, t-тест Уэлча). Заведомо уязвимое сравнение тест ловит (|t| 8–13 тыс.); у сравнения секретов, проверки тега шифрования, проверки ключа API, HMAC и ChaCha20 с разными ключами |t| ниже порога 4.5. Это статистическая проверка, а не доказательство и не замена анализа по питанию и излучению.
- **Разбор HTTP по RFC 9112**: отклоняются пробел перед двоеточием, перенос заголовков, одиночный CR, недопустимые имена и управляющие символы, отсутствие или повтор `Host`. Атаки подмены запросов (CL.TE, TE.CL, CL.CL и др.) напрямую и через nginx 1.27 с постоянными соединениями к synrag — ни один скрытый запрос не дошёл.
- **Внутренний аудит безопасности** (`docs/SECURITY_REVIEW.md`): криптография проверена на наборах Wycheproof (325 случаев ChaCha20-Poly1305, ~500 случаев RSA-подписи, 174 HMAC — все проходят); при чтении кода найдено и исправлено 9 проблем — SSO-токены без проверки аудитории, конфликтующие `Content-Length`, нет предела времени на медленные запросы, утечка частот слов закрытых документов через `termstats`, файлы данных с правами 0644, записи зашифрованного журнала не были привязаны к позиции и снимку, подделка строк текстового лога, выделение памяти по данным повреждённого индекса, разбор флагов (в Helm `--trust-proxy` «съедал» `--acl-default`). Это проверка силами команды, а не независимый аудит.
- **Аудит**: каждая запись содержит хэш предыдущей, `synrag audit-verify` находит удалённые и изменённые строки (проверено на 2 622 записях). **Шифрование хранения**: снимки индекса и каждая запись журнала изменений; чужой ключ или изменённый файл обнаруживаются при загрузке.
- **Устойчивость**: пул потоков с очередью и ответом 503 при перегрузке, постоянные соединения, лимиты на размер запроса, ошибка в одном запросе не роняет сервер. **Парсеры** проверены собственным фаззером: 88 млн искажённых файлов (в том числе реальный PDF 840 КБ) — найдена и исправлена одна ошибка в PDF-парсере; добавлены лимит распаковки архивов (1 ГБ на документ, защита от zip-бомб) и глубины вложенности (4 уровня).
- **Редакции и лицензии** (версии 1.15–1.16): Community — бесплатно, до 1 млн фрагментов на организацию со всем поиском; SSO, права на документы, аудит, шифрование и кластер — по подписанному файлу лицензии, который проверяется без сети (нет сервера активации и телеметрии). Без лицензии запуск с такой функцией отклоняется с понятным сообщением, а сервер без лицензии не открывает данные с правами доступа к документам. После окончания подписки: 30 дней всё работает, затем 90 дней только чтение со всеми функциями защиты, затем сервер не запускается до продления; `synrag export` выгружает все документы с текстом и правами доступа в любой момент, без лицензии. Production — на одну установку (число установок записано в лицензии), Enterprise — на организацию. SDK, MCP и примеры — Apache 2.0, движок и модели — Community License. Скорость не изменилась: сравнение старой и новой сборки вперемешку — кластер пачками 28.5 против 28.1 тыс. док/с (медианы 4 пар, разброс 27.3–29.4 тыс.). **MCP-сервер** (`python -m synrag.mcp`) — поиск и сжатый контекст с цитатами для Claude, Cursor и других агентов; запись — только с явным флагом.
- **Развёртывание**: Docker-образ 27 МБ (distroless: только glibc; прежний образ 2.8 МБ на musl строил базу на ~55 % медленнее — 1 млн отрывков MS MARCO: 71 против 46 с до готовности; качество то же), Helm-чарт (шарды × реплики, координатор, ingress с TLS, проверки готовности, секреты). **Коннекторы** с переносом прав: S3, Confluence, SharePoint / OneDrive, Google Drive, SQL.
- Тесты: 95 на Rust (в том числе лицензии и экспорт, сервер, кластер с отказом и восстановлением реплик, «побеждает последний», права, шифрование и якоря, разбор HTTP, zip-бомбы, наборы Wycheproof) и 11 на Python (SDK, MCP-сервер и все коннекторы на макетах API с реальным сервером synrag).

---

## 9. Качество ответа LLM и экономия токенов

LLM Qwen3-8B (локально), 7 языков MLQA, 300 вопросов на язык, 2100 вопросов всего. Ответ сверяется с эталоном (F1, как в SQuAD). Токены — реальные токены промпта. Доверительные интервалы — парный бутстреп, 95 %.

| Контекст (одни и те же найденные документы) | F1 | EM | Реальных токенов |
|---|---|---|---|
| без контекста | 0.086 | 0.034 | 92 |
| только первый документ, бюджет 250 / 500 | 0.343 / 0.366 | 0.243 / 0.260 | 284 / 346 |
| отбор предложений по эмбеддингам (как LlamaIndex / LangChain), 250 / 500 | 0.327 / 0.365 | 0.231 / 0.260 | 354 / 615 |
| обычный RAG: склейка найденных документов, 250 / 500 | 0.357 / 0.394 | 0.254 / 0.272 | 361 / 624 |
| отбор предложений по BM25, 250 / 500 | 0.387 / 0.415 | 0.274 / 0.296 | 358 / 622 |
| **synrag, 250 / 500** | **0.395 / 0.419** | **0.282 / 0.294** | **357 / 620** |
| правильный абзац (потолок) | 0.655 | 0.473 | 414 |

| Сравнение (одни и те же найденные документы) | Разница F1 | 95 % интервал |
|---|---|---|

| | | |
|---|---|---|
| synrag @250 против склейки @500 (−43 % токенов) | +0.000 | [−0.015; +0.017] |
| synrag против склейки, равный бюджет 250 / 500 | **+0.038 / +0.024** | [+0.021; +0.055] / [+0.009; +0.039] |
| synrag против отбора предложений по эмбеддингам (как LlamaIndex / LangChain) | **+0.068 / +0.053** | [+0.052; +0.085] / [+0.039; +0.068] |
| synrag против «только первый документ» | **+0.052 / +0.053** | [+0.035; +0.070] / [+0.037; +0.069] |
| synrag против отбора предложений по BM25 | +0.008 / +0.004 | [−0.003; +0.018] / [−0.007; +0.014] |

- Против обычного RAG: **сопоставимый ответ при на 43 % меньшем промпте**, или лучший ответ при том же бюджете. Граница допустимого ухудшения заранее не фиксировалась: интервал [−0.015; +0.017] укладывается в ±0.02 F1, но это сопоставимость, а не формальное доказательство неухудшения. Экономия относится к входным токенам одного вызова LLM; полная стоимость работы агента (число вызовов, выходные токены, повторы) не измерялась.
- Простой отбор предложений по BM25 даёт сопоставимый результат: экономию даёт извлечение нужных предложений, а не уникальность алгоритма упаковки. Ценность synrag — в том, что поиск, извлечение и цитаты уже собраны в одном быстром движке для любого языка.
- При бюджете 1000+ токенов разницы со склейкой нет: LLM сама находит ответ в длинном контексте.

---

## 10. Что это значит для компаний

**Малые и средние компании, SaaS, разработчики агентов**

- Не нужны GPU, Elasticsearch-кластер и платный API эмбеддингов: всё работает на одном CPU-сервере или прямо в приложении (память агента, поиск по документам с цитатами).
- База из миллиона документов строится за секунды-минуты, а не часы; переиндексация почти бесплатна.
- Входные токены LLM на RAG-часть — примерно вдвое меньше при том же качестве ответа.

**Крупные компании**

- Обработка больших объёмов эмбеддингов на CPU: по скорости кодирования 100 млн документов размера FiQA (~760 байт) — ~30 мин на одном сервере против ~15 ч у e5-small на GPU с fp16 и ~16 суток у него же на CPU с int8. Это экстраполяция скорости кодирования; полный цикл на таком объёме ещё не проверен.
- Локальная установка без передачи данных наружу; языки СНГ: на русском, казахском, узбекском, азербайджанском, турецком synrag лучше готового стека Tantivy + Model2Vec-RS на 0.02–0.09 nDCG@10 (раздел 7б); EmbeddingGemma на GPU synrag обходит на казахском, узбекском, азербайджанском (+0.03…+0.05) и уступает ей на русском и турецком (−0.02…−0.05).
- Уже есть: обновление базы на лету и журнал для восстановления после сбоя (раздел 8а), 8.8 млн отрывков на одном сервере с приближённым векторным поиском (раздел 8б), а с версии 1.9 — SSO, роли, права на документы, аудит, шифрование, мониторинг, кластер, коннекторы, Docker и Helm (раздел 8д). Чего ещё нет — раздел 11.

**Где выбирать модели эмбеддингов на GPU**: если важен каждый пункт качества на английском (bge-small: +0.023, EmbeddingGemma: +0.048 nDCG@10 к synrag с переранжировщиком) или на хинди, и есть GPU-инфраструктура. Гибридная схема возможна: synrag для быстрого первого отбора и контекста, тяжёлая модель — для переранжирования.

---

## 11. Ограничения (честно)

1. Масштаб проверен до 8.8 млн коротких отрывков (MS MARCO, ~340 байт) на одном сервере (раздел 8б; пик памяти 16 ГБ) и до 2 шардов × 2 реплик на двух машинах (раздел 8д); 100 млн документов и кластеры из многих узлов не проверялись. Векторный поиск на больших базах приближённый (IVF). В кластере нет автоматического перешардирования (на 2 шардах nDCG −0.001…−0.002). Порядок записей от нескольких координаторов — «побеждает последний» по часам координаторов (нужен NTP; внутри расхождения часов побеждает более поздняя метка, а не более поздняя запись) — где это важно, есть условная запись `if_version`. Если один и тот же документ меняют обычной записью на одной стороне разделения сети и любой записью на другой, после слияния остаётся только более позднее изменение. В режиме `write_ack: all` запись при недоступной реплике отклоняется; в `available` — проходит, но до восстановления живёт на одном диске, а условные записи в этом режиме требуют ответа всех реплик шарда.
2. Обновления (раздел 8а): один процесс-писатель; снимок пишется целиком; модель корпуса (`--learn`) на лету не переобучается. Обновления на миллионах документов не проверялись.
3. Оптимизированные модели мерились в типовых вариантах (ONNX Runtime int8 на CPU, fp16 на GPU): ×1.3–2. TensorRT, OpenVINO и дистиллированные/обрезанные модели не проверялись; они могут дать ещё 1.5–3 раза —

разница с Synapse остаётся на два порядка на CPU и в ~10 раз против GPU. Кратности ускорения верны для измеренных конфигураций, а не для любой production-настройки конкурента.

4. Elasticsearch использован с английским analyzer для всех языков, кроме Belebele (раздел 7в, там — встроенные языковые analyzer); языковые плагины улучшат его на китайском и арабском в разделах 7–7б.
5. LLM-тест — одна модель (Qwen3-8B) и вопросы с одним фактом; нужны более крупные LLM и вопросы из нескольких документов. Не сравнивались нейросетевые упаковщики контекста (LLMLingua-2 и др.).
6. Межъязыковой поиск (вопрос и документы на разных языках) пока слабый.
7. На китайском модель корпуса раздувает индекс (117 МБ против 6 МБ у Elasticsearch) из-за n-грамм (в быстром режиме модели корпуса нет).
8. Synapse без BM25 слабее трансформерных моделей и не лучше Model2Vec: на английском на уровне potion-base-8M, на пяти языках хуже potion-multilingual-128M; по скорости Model2Vec-RS не медленнее (английская модель — в 13 раз быстрее). Его сила — работа в паре с BM25 и малая память.
9. Лёгкие стеммеры есть для английского, русского, арабского и хинди; для остальных языков — обрезка основы до 6 букв. Стеммер хинди применяется ко всему письму деванагари: на маратхи (MQA) BM25 −0.029 на dev, с Synapse без потерь. Обрезка 6 вместо 7 (версия 1.4) немного снизила английский BM25 на NFCorpus (−0.009, с Synapse −0.006). LLM-тест (раздел 9) сделан до изменений анализатора версий 1.3–1.4.
10. Переранжировщик (версия 1.7): ~11.5 мс CPU на запрос, пропускная способность ~0.48 тыс. запросов/с на 12 потоках вместо 7–190 тыс.; видит только первые 30 кандидатов (что не нашёл гибрид, он не вернёт); обучен повторять учителя, поэтому лучше EmbeddingGemma стать не может; на MS MARCO (8.8 млн) и в LLM-тесте не проверялся.
11. Гибриды Elasticsearch + модель собраны в типовой настройке (`english` analyzer на всех языках, вес 0.6 без подбора); на 12 языках MQA такой гибрид с EmbeddingGemma хуже одной Gemma (0.445 против 0.486), то есть гибриды конкурентов могут быть недооценены. Одна EmbeddingGemma на MQA в среднем лучше synrag (0.486 против 0.478).
12. Все цифры — внутренние замеры Synlabs, не независимый аудит; скрипты для повторения приведены ниже.
13. Enterprise-функции версии 1.9: криптография (SHA-256, RSA-проверка подписи, ChaCha20-Poly1305) написана на `std` и сверена с тестовыми векторами RFC / NIST / AWS и наборами Wycheproof, прошла статистический тест на утечки по времени, внутренний аудит проведён (`docs/SECURITY_REVIEW.md`, три раунда), но независимого аудита безопасности не было; TLS не реализован в программе намеренно (ingress снаружи, mesh или sidecar между узлами — проверено с ghostunnel); коннекторы проверены на макетах API, а не на реальных SharePoint / Confluence / Google; OCR — внешняя программа (`--ocr-cmd`); старые форматы .doc / .xls / .ppt и почта (.eml, .msg) не поддерживаются.
14. На независимых английских наборах (раздел 7в) synrag ниже bge-small и всех гибридов Elasticsearch + модель на GPU (0.518 против 0.537–0.561); на ArguAna (аргумент → контраргумент) Synapse ухудшает BM25 (0.367 → 0.344). Преимущество на Belebele держится на языках с малыми ресурсами; на хинди, немецком, персидском, турецком EmbeddingGemma лучше на 0.03–0.11.

---

## 12. Как воспроизвести

```
cargo build --release
./target/release/synrag beir DATA --split test --learn --student synapse.synm --set w_dense=0.5 --set
w_global=0.6 --stats
./target/release/synrag beir DATA --split test --student synapse.synm --set w_global=0.6 --rerank rerank.synr   #
+ переранжировщик (вес 0.4, 30 кандидатов)
./target/release/synrag rerank-train --student synapse.synm -o rerank.synr teacher/data2/shard_*.tsv --k 128 --
per-shard 30000   # обучение переранжировщика
python3 bench/es_beir.py DATA test [--analyzer russian]  # Elasticsearch 8.15 в Docker
python3 bench/hf_beir.py scidocs DATA --splits test; python3 bench/belebele_beir.py OUT kaz_Cyrl   # независимые
наборы (7в)
synrag beir DATA --split test --student synapse.synm --set w_global=0.6 --save-index d.syn && synrag serve d.syn
--student synapse.synm --set w_global=0.6 &
python3 bench/http_bench.py DATA test --target synrag --url http://127.0.0.1:7878   # HTTP против HTTP (и --
target es [--keepalive])
synrag serve --data DIR --api-keys keys.json --audit audit.log [--cluster cluster.json]   # enterprise-сервер
(раздел 8д)
python3 bench/load_http.py DATA --url URL --collection kb --api-key K --acl-groups 10      # загрузка с правами
доступа
cargo run --release --example fuzz -- tests/fuzz_seeds 300                                  # фаззинг парсеров
python bench/dense_beir.py DATA test --model intfloat/multilingual-e5-small --device cuda
python bench/onnx_bench.py --model intfloat/multilingual-e5-small --speed-data FIQA --quality-data SCIFACT   #
ONNX int8 / fp16
python bench/m2v_beir.py DATA test --model minishlab/potion-multilingual-128M                    # Model2Vec
(Python)
(cd bench/tantivy_m2v && RUSTFLAGS="-C target-cpu=native" cargo build --release)
bench/tantivy_m2v/target/release/tantivy_m2v DATA test --model minishlab/potion-multilingual-128M --lang ar   #
Tantivy + Model2Vec-RS
```

```
synrag beir DATA --split test ... --dump-contexts ctx.jsonl --baselines   # контексты для LLM
python3 bench/llm_answer.py DATA ctx.jsonl --url http://127.0.0.1:11434 --model qwen3-8b --sample 300
python3 bench/llm_compare.py scores*.jsonl --pairs pack@250:naive@500
```

Журнал всех экспериментов, включая неудачные: `EXPERIMENTS.md` (E1–E52).

**Изменения в версии 1.1.** Время холодного старта synrag теперь замеряется внутри программы от запуска до готовности (в 1.0 стояло 3.6 с — без загрузки и прогона Synapse; верно 4.8 с для полного режима). Добавлены быстрый режим (BM25 + Synapse, 1.6 с), оптимизированные модели эмбеддингов (ONNX int8, fp16), Model2Vec, Synapse отдельно, оговорки об экстраполяции на 100 млн документов и о границах вывода по токенам.

**Изменения в версии 1.2.** Добавлено сравнение с готовым стеком на Rust (Tantivy + Model2Vec-RS, раздел 7а). Исправлено утверждение версии 1.1 «Synapse в 2–3 раза быстрее Model2Vec»: оно верно только против Python-версии; официальная Rust-версия Model2Vec не медленнее Synapse (английская модель — в 13 раз быстрее).

**Изменения в версии 1.3.** В анализатор synrag добавлены лёгкие стеммеры для арабского (Light10) и хинди (Ramanathan & Rao), выбор настроек — на dev. MLQA test: арабский 0.517 → 0.578, хинди 0.586 → 0.607 (быстрый режим); среднее по пяти языкам 0.634 → 0.650 (быстрый), 0.635 → 0.651 (полный). Формат индекса v3: индексы, построенные прежними версиями, нужно перестроить.

**Изменения в версии 1.4.** Обрезка основы до 6 букв для латиницы (было 7) и после русского стеммера для кириллицы (было без обрезки); выбор — на dev. Новый раздел 7б: 12 языков MQA, включая турецкий, казахский, узбекский, азербайджанский. Формат индекса v4.

**Изменения в версии 1.5.** Обновление базы без перестройки: добавление, замена и удаление документов на лету (CLI, HTTP, библиотека), раздел 8а.

**Изменения в версии 1.6.** Журнал изменений для восстановления после сбоя (8а); масштаб 8.8 млн документов (MS MARCO), снижение пиковой памяти и приближённый векторный поиск IVF (8б); статический Synapse — турбо-режим кодирования (раздел 5).

**Изменения в версии 1.7.** Контекстный переранжировщик поверх Synapse (`--rerank`): позднее взаимодействие токенов вопроса и документа на первых 30 кандидатах, обучен по ранжированию учителя; вес и число кандидатов выбраны на dev. Test: BEIR 0.441 → 0.451 (быстрый), 0.444 → 0.455 (полный); MLQA 0.651 → 0.660 и 0.653 → 0.664; MQA 12 языков 0.460 → 0.478. Цена — запрос ~3 мс и пропускная способность ниже в ~24 раза (разделы 6–8, 11). Добавлены сводная таблица «все против всех» (1а), Elasticsearch и модели эмбеддингов на 12 языках MQA (7б; EmbeddingGemma там в среднем сильнее synrag), раздел о больших данных (8в), стоимость индексации в долларах (8г), сравнение HTTP против HTTP с Elasticsearch (8), Python SDK и пример агента (2).

**Изменения в версии 1.8.** Независимые наборы (раздел 7в): BEIR SciDocs, ArguAna, TREC-COVID, Quora и Belebele на 14 языках (английский, русский, украинский, казахский, узбекский, азербайджанский, киргизский, таджикский, турецкий, немецкий, арабский, персидский, хинди, китайский) — synrag, Elasticsearch (на Belebele — с языковыми analyzer), Tantivy + Model2Vec-RS, модели эмбеддингов и гибриды, без подбора настроек. Итог честный в обе стороны: на английском synrag ниже bge-small и гибридов, на Belebele в среднем выше EmbeddingGemma за счёт языков с малыми ресурсами.

**Изменения в версии 1.9.** Enterprise-функции (раздел 8д, `docs/ENTERPRISE.md`): коллекции, ключи API и SSO-токены (OIDC), роли, права доступа на каждый документ, журнал аудита с цепочкой хэшей, шифрование хранения, метрики Prometheus, JSON-логи, пул потоков с защитой от перегрузки, загрузка файлов по HTTP, кластер (шарды, реплики, координатор), коннекторы (S3, Confluence, SharePoint, Google Drive, SQL, папки), Docker и Helm, фаззинг парсеров и лимиты архивов. Разделы 1, 2, 8, 8в, 10, 11 обновлены: пробелы «нет распределённого режима, прав доступа, аудита, коннекторов» закрыты; новые ограничения — в разделе 11 (пункт 14).

**Изменения в версии 1.10.** Проверка замечаний к 1.9 (раздел 8д): качество с ключом API равно качеству без авторизации (0.2788; разница в первой редакции — настройка нарезки и двойной учёт повторов в скрипте замера); поиск в кластере в две фазы с общей статистикой слов и общими границами нормировки — потеря на 2 шардах уменьшилась с −0.007 до −0.001. Внутренний аудит безопасности (`docs/SECURITY_REVIEW.md`): наборы Wycheproof, 9 исправленных проблем. Кластер на двух машинах с отказами реплики и зависшим узлом — без ошибок; восстановление отставшей реплики `/cluster/repair`.

**Изменения в версии 1.11.** Второй раунд по замечаниям к 1.10 (раздел 8д): разбор HTTP строго по RFC 9112 и проверка атак подмены запросов через nginx; версии записей «побеждает последний», режимы подтверждения записи, автоматическое восстановление отставших реплик; внешние контрольные точки аудита, якорь состояния против отката данных, ротация ключа; обязательное объявление защиты канала между узлами, mTLS через mesh / sidecar (проверено на двух машинах), постоянные соединения координатора с шардами (p99 между машинами 101 → 29 мс).

**Изменения в версии 1.12.** Третий раунд по замечаниям к 1.11: версии записей уникальны для каждого координатора; условная запись `if_version`; восстановление реплик слиянием в обе стороны (изменения разных документов с обеих сторон разделения сети сохраняются; для одного документа — более позднее); контракт сохранности подтверждённой записи; якорь состояния синхронизируется на каждую запись по умолчанию; `rekey` безопасен при повторе;

статистический тест на утечки по времени; в `SECURITY_REVIEW.md` исправлен раздел остаточных рисков (ротация ключа уже есть).

**Изменения в версии 1.13.** По замечаниям к 1.12: условная запись атомарна во всём кластере (решает первичная реплика шарда; без неё — отказ 503); версия документа в результатах поиска; адрес клиента из `X-Forwarded-For` только от доверенных прокси и справа налево (раньше клиент мог подделать адрес в аудите); проверено обновление с версии 1.8; замерена скорость записи со всеми гарантиями; процедуры обновления и восстановления из копии при внешнем якоре.

**Изменения в версии 1.14.** По замечаниям к 1.13 (раздел 8д): по сценарию рецензента найдена и исправлена ошибка — вернувшаяся после разделения сети первичная реплика могла принять условную запись по своей устаревшей копии в режиме `write_ack: available` (теперь условие проверяется по самой новой версии среди всех реплик; тест воспроизводил ошибку и теперь проходит); `node_id` координатора обязателен, его уникальность проверяется арендой на шардах (дубликат не запускается), Helm запускает координаторы как StatefulSet с номером из имени пода; исправлена слишком сильная формулировка о слиянии после разделения сети (для одного документа остаётся более позднее изменение); тест конкурентных условных записей расширен с 20 до 400 раундов в обоих режимах; скорость записи перемерена с повторами, сравнение старой и новой сборки — без замедления. Поиск не менялся.

**Изменения в версии 1.15.** Редакции: бесплатная Community (один сервер, до 1 млн фрагментов, весь поиск) и лицензионные функции (SSO, права на документы, аудит, шифрование, кластер) по подписанному файлу лицензии с проверкой без сети; истечение лицензии не блокирует данные. MCP-сервер для агентов. Поиск не менялся, скорость записи — та же. Исправление: Docker-образ 2.8 МБ собирался под musl, и его распределитель памяти замедлял построение базы на ~55 % (1 млн отрывков MS MARCO: 71 против 46 с, пропускная способность запросов −9 %); образ теперь на glibc (distroless, 27 МБ), а готовый бинарник для загрузки собран под glibc 2.30+ (Ubuntu 20.04+, Debian 11+) — та же скорость, что в замерах отчёта. Сайт ai.synlabs.pro: документация, бенчмарки, загрузка, установка одной командой.

**Изменения в версии 1.16.** Условия лицензии без лазеек: бесплатный лимит 1 млн фрагментов считается на организацию (вместе с дочерними компаниями, дробление запрещено), Production покрывает одну установку на единицу подписки, после окончания подписки — 30 дней полной работы, 90 дней только чтения, затем остановка обслуживания; новая команда `synrag export` выгружает данные всегда, без лицензии; сервер без лицензии не открывает данные с правами доступа. Цены вынесены из лицензии на страницу тарифов. Python SDK, MCP-сервер и примеры переведены на Apache 2.0. Публичный репозиторий github.com/Synlabs-ai/synrag (README, бенчмарки, SDK, Docker, релизы; исходники движка не публикуются).