

Статья поступила в редакцию: 12.09.2026

Статья принята к публикации: 18.09.2026

Статья опубликована: 28.09.2026

УДК 004.89+004.4

DOI: 10.32743/UniTech.2026.150.9.23398

Сравнение эмбединг-моделей для семантического кэширования запросов к большим языковым моделям

Колышкин Виктор Викторович

инженер-программист

независимый исследователь

Российская Федерация, г. Москва

ORCID: 0009-0005-2632-5107

E-mail: mr.viktor.kolyshkin@gmail.com

A comparison of embedding models for semantic caching of large language model requests

Kolyshkin Viktor

Software engineer, independent researcher

Russia, Moscow

Аннотация

Обращения к большим языковым моделям (LLM) через API дороги и медленны, а часть запросов повторяется по смыслу; семантический кэш возвращает готовый ответ на близкий запрос, снижая стоимость и задержку. Ключевой выбор — эмбединг-модель и порог сходства — управляет скрытым компромиссом между долей попаданий и долей ложных попаданий (кэш вернул ответ на другой запрос). В работе на публичном наборе Quora Question Pairs (40 430 пар) сравниваются три компактные модели (all-MiniLM-L6-v2, e5-small-v2, bge-small-en-v1.5) и два лексических базиса (точное совпадение, TF-IDF) по совместным осям: ROC/AUC, доля попаданий при заданном бюджете ложных попаданий, латентность и пропускная способность на CPU и GPU, а также оценка денежной экономии. Семантические модели (AUC 0,86–0,88) заметно превосходят лексические (TF-IDF 0,73; точное совпадение ловит менее 0,5 % смысловых дублей). Модель bge-small даёт лучшее качество, но уступает MiniLM по скорости на CPU; при строгом бюджете ложных попаданий (1 %) в попарной оценке отделяется лишь около одной шестой повторов. Стоимость эмбединга пренебрежимо мала относительно экономии на вызовах LLM. Отдельно

Библиографическое описание: Колышкин В.В. Сравнение эмбединг-моделей для семантического кэширования запросов к большим языковым моделям // Universum: технические науки : электрон. научн. журн. 2026. 9(150). URL: <https://7universum.com/ru/tech/archive/item/23398>

показано, что операционная доля ложных попаданий растёт с размером кэша и превышает попарную оценку. Результаты дают инженеру основу для выбора модели и порога под целевой профиль.

Abstract

Calls to large language models (LLMs) via API are costly and slow, while a share of requests repeats in meaning; a semantic cache returns a stored answer for a similar request, cutting cost and latency. The key choice — the embedding model and the similarity threshold — governs a hidden trade-off between the hit rate and the false-hit rate (a cached answer served for a different request). Using the public Quora Question Pairs set (40,430 pairs), we compare three compact models (all-MiniLM-L6-v2, e5-small-v2, bge-small-en-v1.5) and two lexical baselines (exact match, TF-IDF) along joint axes: ROC/AUC, hit rate at a fixed false-hit budget, latency and throughput on CPU and GPU, and an estimate of monetary savings. Semantic models (AUC 0.86–0.88) substantially outperform lexical ones (TF-IDF 0.73; exact match catches under 0.5 % of semantic duplicates). bge-small gives the best quality but trails MiniLM in CPU speed; at a strict false-hit budget (1 %) only about one sixth of repeats is separated in the pairwise setting. Embedding cost is negligible relative to the savings on LLM calls. We further show that the operational false-hit rate grows with cache size and exceeds the pairwise estimate. The results give an engineer a basis for choosing a model and threshold for a target profile.

Ключевые слова: семантическое кэширование, большие языковые модели, эмбединги предложений, косинусное сходство, ROC-анализ, ложные попадания, оптимизация стоимости.

Keywords: semantic caching, large language models, sentence embeddings, cosine similarity, ROC analysis, false hits, cost optimization.

Введение

Обращения к большим языковым моделям (LLM) через облачные API стали типовым компонентом веб-систем, но каждый вызов стоит денег и добавляет сетевую задержку. На практике, часть пользовательских запросов повторяется по смыслу, отличаясь лишь формулировкой. Семантический кэш перехватывает такой повтор: он кодирует запрос в вектор, ищет ближайший среди ранее обслуженных и, если сходство превышает порог, возвращает готовый ответ, не обращаясь к модели [1]. Такой подход снижает и стоимость, и задержку, и уже реализован в открытых системах (GPTRCache [1], MeanCache [6]) и как одна из стратегий сокращения расходов на LLM [3].

Семантический кэш держится на двух устоявшихся компонентах. Первый компонент, векторные представления предложений (от Sentence-BERT [10] до компактных дистиллированных и контрастно обученных моделей MiniLM [13], E5 [12], BGE [14]), кодирует смысл короткого текста для сравнения косинусом. Второй, приближённый поиск ближайшего соседа (FAISS [7], HNSW [8]), обеспечивает масштабирование. Эти компоненты мы не изобретаем, а применяем и сравниваем.

Однако у кэша есть скрытая цена ошибки. При слишком низком пороге сходства кэш вернёт ответ на другой запрос; такое ложное попадание тихо подменяет смысл. Существующие системы [1; 6] фиксируют эмбеddинг-модель де-факто, а универсальные бенчмарки эмбеddеров (МТЕВ [9], STS [2]) дают агрегатные метрики качества (в том числе на классификации пар), но не измеряют операционный профиль кэша, где на одной оси сходятся доля попаданий при фиксированном бюджете ложных попаданий, выбор порога по ROC и стоимость (latency на CPU/GPU и деньги). Недавние работы затрагивают эту задачу — дообучение доменных эмбеddеров и ансамблирование моделей для кэша [4; 5], — но систематического воспроизводимого сравнения открытых эмбеddеров под семантический кэш, по этим совместным осям, в рецензируемой литературе, насколько нам известно, пока недостаёт; вендорные сравнения встречаются, но не рецензируются и не публикуют воспроизводимую методику.

Мотивация этой работы — практическая. Работая над production-конвейером, который генерирует веб-контент из внешних источников с помощью облачных LLM, мы наблюдали, что одна задача порождает порядка восьмидесяти обращений к модели, и заметная их часть оказывалась почти одинаковой по смыслу, по сути перефразировками. При этом провайдерский prompt-кэш на такой неравномерной нагрузке фактически не срабатывал, а дедупликация выполнялась лишь по точному совпадению ключа, а не по смыслу. Так и возник вопрос, насколько семантическое сопоставление запросов способно перехватить эти повторы и какой ценой.

Цель работы — восполнить этот пробел. Для этого на публичном наборе Quora Question Pairs (QQP) [11] мы сравниваем три компактные эмбеddинг-модели и два лексических базиса, решая задачи: (1) оценить качество разделения дубликатов и не-дубликатов (ROC/AUC); (2) измерить долю попаданий при фиксированном бюджете ложных попаданий; (3) измерить латентность и пропускную способность кодирования на CPU и GPU; (4) измерить латентность поиска в зависимости от размера кэша; (5) оценить денежную экономию. Вклад работы — воспроизводимая методика и количественные ориентиры для выбора модели и порога под целевой профиль нагрузки.

Материалы и методы

Постановка. Пара текстов из QQP имеет метку «дубликат» (вопросы эквивалентны по смыслу) или «не дубликат». Мы трактуем метку как эталон работы кэша: пришёл запрос q_2 , в кэше лежит q_1 ; кэш кодирует оба, считает косинусное сходство s и при $s \geq \theta$ возвращает ответ, связанный с q_1 (попадание). Тогда при пороге θ :

истинное попадание (true hit): $s \geq \theta$ и пара — дубликат (кэш корректно переиспользовал ответ);

ложное попадание (false hit): $s \geq \theta$ и пара — не дубликат (кэш вернул ответ на другой вопрос);

промах (miss): $s < \theta$ (запрос уходит в LLM).

Доля попаданий (hit rate) равна доле верно перехваченных дубликатов (TPR), доля ложных попаданий (false-hit rate) — доле ошибочно перехваченных не-дубликатов (FPR). Изменение θ задаёт ROC-кривую; площадь под ней (AUC) характеризует разделяющую способность модели независимо от выбора порога.

Данные. Набор Quora Question Pairs (в составе GLUE [11]); использован размеченный раздел validation (40 430 пар), из них 14 885 (36,8 %) помечены как дубликаты. Метки взяты как есть, без дополнительной очистки.

Сравниваемые модели. Три компактные эмбединг-модели с открытыми весами, все с размерностью вектора 384: all-MiniLM-L6-v2 [13], e5-small-v2 [12] (с рекомендованным префиксом «query:»), bge-small-en-v1.5 [14]. Два лексических базиса: точное совпадение нормализованных строк (нижний регистр, удаление пунктуации) и косинус по TF-IDF-векторам, обученным на всех вопросах набора.

Метрики и стенд. Качество — AUC и рабочие точки (порог и hit rate при бюджете ложных попаданий 1 %, 5 %, 10 %). Латентность кодирования — медиана по трём повторам после прогрева, отдельно на CPU и GPU, на подвыборке 3 000 запросов; отсюда пропускная способность (запросов/с). Абсолютные тайминги сняты на одном десктопе и зависят от фоновой нагрузки, поэтому читаются как индикативные; устойчив здесь относительный порядок, а не точное значение. Латентность поиска — медианное время brute-force поиска ближайшего среди N ранее сохранённых 384-мерных векторов (эмбединги bge) при росте N , по 1 000 запросов с прогревом. Модель e5 по рекомендации авторов кодирует оба текста пары с префиксом «query:» (симметричный режим). **Операционная проверка** (модель bge): порог θ калибруется на отдельном calibration-сплите (половина пар) под pairwise-FPR, после чего на независимом test-сплите строится store из N записей и при поиске ближайшего соседа по всему store при росте N измеряются recall и операционный false-hit. Экономия — модель на основе доли повторяемых запросов и recall кэша (см. Результаты). Стенд: Python 3.10, PyTorch 2.11 (CUDA 12.8), sentence-transformers 3.3; CPU и GPU NVIDIA GeForce RTX 5070 Ti; фиксированный seed = 42; код и конфигурации приложены для воспроизводимости. Оценки AUC получены на одном фиксированном разбиении (validation), без доверительных интервалов.

Результаты и обсуждение

Качество разделения

Таблица 1.

Разделяющая способность моделей на QQP (validation)

Модель	Размерность	AUC
bge-small-en-v1.5	384	0,876
all-MiniLM-L6-v2	384	0,871
e5-small-v2	384	0,864
TF-IDF (базис)	—	0,733
точное совпадение (базис)	—	ловит 67 из 14 885 дублей

Три семантические модели близки по AUC (0,864–0,876) и заметно превосходят лексику; TF-IDF даёт 0,733, а точное совпадение строк перехватывает лишь 67 из 14 885 смысловых дубликатов (менее 0,5 %). Смысловые повторы почти никогда не совпадают дословно, поэтому наивный кэш по точному совпадению практически бесполезен, а лексическое сходство (TF-IDF) отстаёт от семантического примерно на 0,14 AUC. На рисунке 1 кривые эмбеддеров лежат заметно выше диагонали и выше TF-IDF, а bge-small стабильно проходит выше остальных.

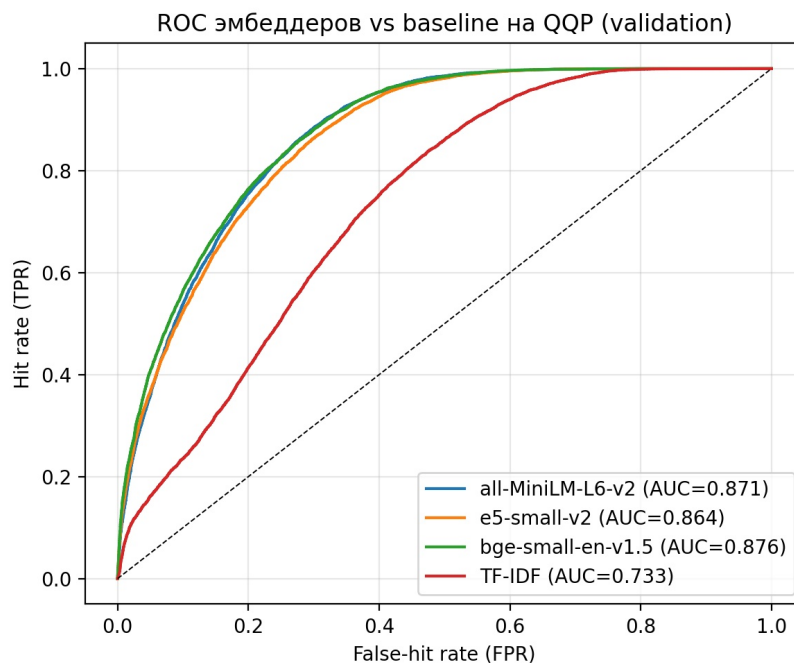


Рисунок 1. ROC-кривые эмбеддеров и базиса TF-IDF на наборе QQP (validation)

Компромисс «попадания против ложных попаданий»

Таблица 2.

Доля попаданий при фиксированном бюджете ложных попаданий

Модель	$\theta@1\%$	hit@1 %	$\theta@5\%$	hit@5 %	$\theta@10\%$	hit@10 %
bge-small-en-v1.5	0,970	0,176	0,930	0,408	0,902	0,564
all-MiniLM-L6-v2	0,965	0,137	0,915	0,359	0,871	0,539
e5-small-v2	0,987	0,153	0,971	0,367	0,959	0,522
TF-IDF	0,943	0,067	0,860	0,162	0,803	0,236

Величины в таблице 2 попарные, то есть false-hit измерен на одной паре. Порог здесь задаёт рабочую точку осознанно, а не «ставится повыше». При попарном бюджете ложных попаданий 1 % даже лучшая модель (bge) отделяет лишь $\approx 18\%$ повторов, при 5 % — $\approx 41\%$, и только при 10 % — свыше половины. Безопасность кэша (мало чужих ответов) и его полезность прямо противоречат друг другу, поэтому рабочую точку выбирают под задачу. Модель bge доминирует на всех бюджетах и по качеству предпочтительна. Абсолютные пороги моделей несопоставимы (у e5 рабочие пороги $\approx 0,96$ – $0,99$, у MiniLM $\approx 0,87$ – $0,97$), поэтому порог калибруют под конкретную модель, а не переносят между ними.

Операционная проверка: false-hit растёт с размером кэша

Попарные величины таблицы 2 недооценивают риск реального кэша. В нём запрос сравнивается со всем множеством из N записей, и ложное попадание наступает, если порог превысит хотя бы один не-дубликат (максимум сходства по N), поэтому операционный false-hit при том же пороге обязан расти с размером кэша. Мы проверяем это напрямую на модели bge. Порог θ калиброван на отдельном calibration-сплите под pairwise-FPR 5 % ($\theta = 0,929$), после чего на независимом test-сплите измерены recall и операционный false-hit при поиске ближайшего по store растущего размера.

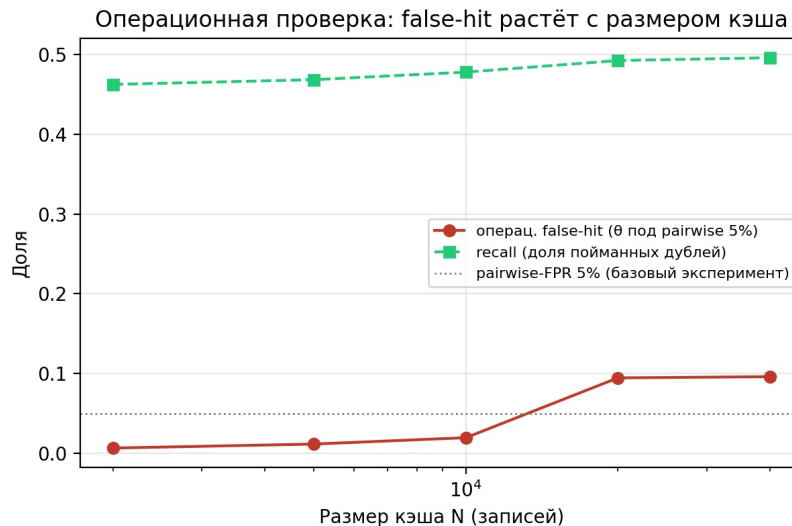


Рисунок 2. Операционный false-hit и recall в зависимости от размера кэша (порог калиброван под pairwise-FPR 5 %)

Механизм подтверждается (рисунок 2). При пороге, дающем 5 % на паре, операционный false-hit растёт с 0,7 % при 2 000 записей до 9,6 % при 40 000 — почти вдвое выше номинального бюджета, тогда как recall держится на уровне 0,46–0,50 (он даже выше попарного, потому что запрос может совпасть с любой записью store, а не только со своей парой). На практике это значит, что попарные пороги таблицы 2 задают лишь нижнюю оценку операционного риска; для большого кэша порог приходится ужесточать под целевой операционный false-hit, а калибровать под реальный размер store.

Стоимость: латентность и пропускная способность

Таблица 3.

Латентность кодирования и пропускная способность (подвыборка 3 000)

Модель	CPU, мс/запрос	CPU, запр./с	GPU, мс/запрос	GPU, запр./с
all-MiniLM-L6-v2	0,74	1343	0,11	9363
bge-small-en-v1.5	1,35	744	0,14	7380
e5-small-v2	1,45	692	0,14	6990

Качество даётся не бесплатно. bge-small на CPU почти вдвое медленнее MiniLM (1,35 против 0,74 мс), а MiniLM при небольшой потере AUC (0,871 против 0,876) даёт наибольшую пропускную способность на CPU. Перенос на GPU (RTX 5070 Ti) ускоряет кодирование примерно в 7–10 раз (MiniLM 0,74 → 0,11 мс, свыше 9 000 запросов/с) и практически снимает стоимость эмбединга как узкое место при высокой нагрузке. Значит, выбор модели зависит от бюджета железа. На CPU-хостинге MiniLM даёт лучший компромисс «качество/скорость», а на GPU разрыв между моделями по скорости сжимается, и можно брать более качественную bge.

Латентность поиска и масштаб кэша

Стоимость перебора зависит от размера кэша. На малых кэшах она определяется постоянными накладными расходами ($\approx 7\text{--}12$ мкс до 1 000 записей), затем растёт с числом записей до ≈ 400 мкс при 40 000 (рисунок 3). Рост не строго пропорционален N из-за эффектов BLAS и иерархии памяти, но при больших кэшах перебор становится сопоставим со стоимостью самого кодирования и мотивирует переход к приближённому индексу (HNSW [8], FAISS [7]). Для кэшей до нескольких тысяч записей перебор приемлем.

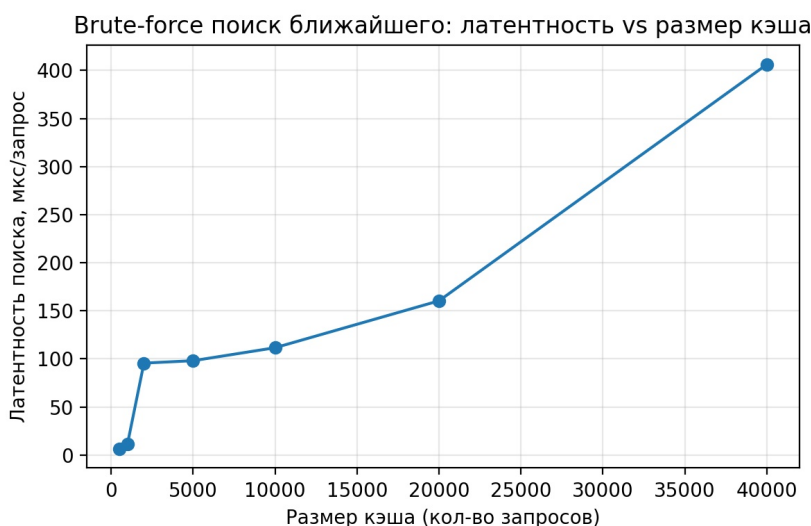


Рисунок 3. Латентность brute-force поиска ближайшего в зависимости от размера кэша

Экономия

Экономия оценим параметрической моделью. Пусть доля повторяемых по смыслу запросов равна r , а recall кэша — доля перехваченных повторов при выбранном бюджете ложных попаданий. Тогда на объём запросов N кэш перехватывает $N \cdot r \cdot \text{recall}$ вызовов к LLM — это измеряемая величина, не зависящая от тарифа модели. Значение r зависит от приложения; для ориентира возьмём $r \approx 31\%$, сообщённое для реального трафика в MeanCache [6], и покажем диапазон 20–50 %. Модель bge при бюджете ложных попаданий 5 % (recall 0,408) даёт:

Таблица 4.

Доля запросов, обслуженных из кэша (модель bge, recall 0,408), на 1 млн запросов

Повторяемость r	Перехвачено вызовов	Доля всех запросов
20 %	81 540	8,2 %
31 % (MeanCache [6])	126 387	12,6 %
50 %	203 850	20,4 %

В деньгах экономия зависит от тарифа. При публичных ценах API (OpenAI и Anthropic, август 2026 г.) и запросе ≈ 1000 входных / 500 выходных токенов перехват 126 387 вызовов на 1 млн запросов ($r = 31\%$) экономит ориентировочно ≈ 57 \$ на GPT-4o-mini, ≈ 442 \$ на Claude Haiku 4.5, ≈ 948 \$ на GPT-4o и ≈ 1327 \$ на Claude Sonnet 4.6; экономия линейна по цене запроса и объёму.

Стоимость самого кодирования пренебрежимо мала относительно вызова LLM (эмбединг на CPU — доли миллисекунды локально, вызов LLM — центры плюс сетевой круг), поэтому чистая экономия практически совпадает с валовой. Кэш «почти бесплатен» относительно того, что экономит; ограничителем выступает recall при допустимом уровне ложных попаданий, а не стоимость эмбединга.

Заключение

Мы сравнили компактные эмбединг-модели для семантического кэширования LLM-запросов по совместному профилю качества, стоимости и экономии на публичном наборе QQR. Семантические модели заметно превосходят лексические базисы, а среди них bge-small даёт лучшее качество на всех бюджетах ложных попаданий, уступая MiniLM в скорости на CPU. Главный практический вывод касается порога сходства. Это осознанный выбор рабочей точки между безопасностью и полезностью кэша (при 1 % ложных попаданий перехватывается лишь $\approx 1/6$ повторов), и калибровать его нужно под конкретную модель. Поскольку стоимость эмбединга пренебрежима относительно экономии, основной инженерный рычаг — выбор модели и порога под целевой профиль, а не скорость модели.

Ограничения. QQR — контролируемый прокси; это пары вопросов-парафразов, а не реальный трафик запросов к LLM, поэтому абсолютные пороги и доли повторов в продакшене будут иными; методика (оси и способ выбора рабочей точки) переносится, конкретные числа требуют калибровки на своих данных. Операционный false-hit измерен в предположении, что негативные пробы не имеют неразмеченного дубликата в store; неполнота разметки QQR может приводить к завышению его абсолютного значения, но тренд роста с размером кэша от этого не меняется. Дальнейшая работа — проверка на втором домене данных и сравнение brute-force поиска с приближённым индексом (HNSW/FAISS) по латентности и точности.

Список литературы

1. Bang F. GPTCache: An Open-Source Semantic Cache for LLM Applications Enabling Faster Answers and Cost Savings // Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS). P. 212–218. – Singapore: Association for Computational Linguistics, 2023. – DOI: 10.18653/v1/2023.nlposs-1.24.
2. Cer D., Diab M., Agirre E. [и др.] SemEvalTask 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation // Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval). P. 1–14. – Vancouver: Association for Computational Linguistics, 2017. – DOI: 10.18653/v1/S17-2001.
3. Chen L., Zaharia M., Zou J. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance [Электронный ресурс] // arXiv preprint. – 2023. – DOI: 10.48550/arXiv.2305.05176.
4. Ghaffari S., Bahranifard Z., Akbari M. An Ensemble Embedding Approach for Improving Semantic Caching Performance in LLM-based Systems [Электронный ресурс] // arXiv preprint. – 2025. – DOI: 10.48550/arXiv.2507.07061.
5. Gill W., Cechmanek J., Hutcherson T. [и др.] Advancing Semantic Caching for LLMs with Domain-Specific Embeddings and Synthetic Data [Электронный ресурс] // arXiv preprint. – 2025. – DOI: 10.48550/arXiv.2504.02268.
6. Gill W., Elidrisi M., Kalapatapu P. [и др.] MeanCache: User-Centric Semantic Caching for LLM Web Services [Электронный ресурс] // arXiv preprint. – 2024. – DOI: 10.48550/arXiv.2403.02694.
7. Johnson J., Douze M., Jégou H. Billion-Scale Similarity Search with GPUs // IEEE Transactions on Big Data. – 2019. – Т. 7. – DOI: 10.1109/TBDDATA.2019.2921572.
8. Malkov Yu. A., Yashunin D.A. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2020. – Т. 42. – DOI: 10.1109/TPAMI.2018.2889473.
9. Muennighoff N., Tazi N., Magne L., Reimers N. MTEB: Massive Text Embedding Benchmark // Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL). P. – Dubrovnik: Association for Computational Linguistics, 2023. – DOI: 10.18653/v1/2023.eacl-main.148.
10. Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks // Proceedings of the. P. 3982–3992. – Hong Kong: Association for Computational Linguistics, 2019. – DOI: 10.18653/v1/D19-1410.
11. Wang A., Singh A., Michael J. [и др.] GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding // Proceedings of the. P. 353–355. – Brussels: Association for Computational Linguistics, 2018. – DOI: 10.18653/v1/W18-5446.
12. Wang L., Yang N., Huang X. [и др.] Text Embeddings by Weakly-Supervised Contrastive Pre-training [Электронный ресурс] // arXiv preprint. – 2022. – DOI: 10.48550/arXiv.2212.03533.

-
13. Wang W., Wei F., Dong L. [и др.] MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers // Advances in Neural Information Processing Systems (NeurIPS). – 2020. – Т. 33.
 14. Xiao S., Liu Z., Zhang P., Muennighoff N. C-Pack: Packed Resources For General Chinese Embeddings [Электронный ресурс] // arXiv preprint. – 2023. – DOI: 10.48550/arXiv.2309.07597.