ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет Санкт-Петербургская школа физико-математических и компьютерных наук

Малашенко Борис Тарасович

Разработка универсального семантического энкодера текстов для русского языка

Выпускная квалификационная работа
по направлению подготовки 01.04.02
Прикладная математика и информатика
образовательная программа «Машинное обучение и анализ данных»

Научный руководитель:
к.ф.-м.н
доцент департамента информатики

Мухин М. С.

Соруководитель:
старший программист
«ВКонтакте»

Спирин Е. С.

Рецензент:
Младший инженер-исследователь
«Одноклассники»

Лебедева А. Ю.

Санкт-Петербург
2024

SAINT-PETERSBURG ACADEMIC UNIVERSITY
Higher education centre

St. Petersburg School of Physics, Mathematics and Computer Science

Malashenko Boris

Universal Semantic Encoder for Russian

Master's Thesis

Research Supervisor:
Associate Professor, Ph. D

Mukhin Mikhail

Co-supervisor:
Senior Programmer «VK»

Spirin Egor

Reviewer:
Junior Research Engineer «OK.ru»

Lebedeva Anna

Saint-Petersburg
2024

Оглавление

Введение

В современном мире, характеризующемся огромными объемами информации и быстрым обменом данных, задачам обработки естественного языка (NLP) уделяется все большее внимание. Одной из ключевых задач в области NLP является кодирование текстов [10], то есть представление их в форме, удобной для алгоритмов машинного обучения, например, в виде набора чисел.

Под семантическим кодированием понимается извлечение смысла текста. Энкодер текстов переводит каждый документ в корпусе в скрытое пространство векторов фиксированной размерности, называемых эмбеддингами. Поверх матрицы эмбеддингов можно применять методы классического машинного обучения, такие как k ближайших соседей, логистическую регрессию и случайные леса [37], а также обучать линейный полносвязный слой. Это возможно благодаря тому, что энкодеры обучаются так, чтобы в скрытом пространстве представления близких по смыслу текстов находились близко друг к другу, а далеких — далеко. Из-за ограниченной рамерности скрытого пространства, модель не может просто запомнить все тексты, а вынуждена выделять скрытые в них признаки. Таким образом, семантические энкодеры создают представление текста, которое сохраняет его смысловую структуру, что позволяет эффективно применять их для различных задач обработки естественного языка. Стоит отметить, что энкодер текстов может не нуждаться в дообучении под конкретную задачу и использоваться «из коробки». Исследования [36, 48] показали, что при достаточно большом наборе данных в обучающей выборке, модель может эффективно справляться даже с задачами, которые она не видела во время обучения. Универсальные семантические энкодеры текстов находят широкое применение в различных областях обработки естественного языка (NLP). Они используются для выполнения множества различных задач, таких как классификация текстов [50], веб поиск [41], суммаризация [46], кластеризация [47], выделение признаков [51] и многих других. Одной из наиболее популярных задач, в которой энкодер текстов является важным компонентом, является Retrieval-Augmented Generation (RAG) [42] — техника, которая повышает производительность

больших языковых моделей путем интеграции механизмов извлечения информации.

Энкодеры текстов могут быть моноязычными, тогда их цель в извлечении векторных представлений только для одного целевого языка. Если же энкодер может работать с несколькими языками, его назвают мультиязычным. Для оценки энкодеров текстов используют бенчмарки – совокупности различных задач. Несмотря на то, что новейшие мультиязычные модели показывают хорошие результаты, таблицы лидеров для английского и китайского языков демонстрируют[1], что моноязычный энкодер текстов может показывать более высокое качество, чем мультиязычный. В настоящее время существует пробел в разработке русскоязычных энкодеров.

Постановка задачи

Целью данной работы является разработка семантического энкодера текстов для русского языка, который покажет более высокое качество, чем существующие модели. Для достижения этой цели были поставлены следующие задачи:

1. Провести обзор предметной области.
2. Разработать алгоритм обучения универсального семантического энкодера, содержащий новейшие подходы, подготовить данные для обучения.
3. Обучить модель по разработанной методологии.
4. Апробировать полученную модель, сравнить с существующими решениями.

[1] https://huggingface.co/spaces/mteb/leaderboard

1. Обзор предметной области

Основная цель выпускной квалификационной работы заключается в разработке передового семантического энкодера текстов для русского языка. Для достижения цели в данном исследовании проводится глубокий анализ существующих методик обучения подобных моделей. Важным этапом является выбор оптимального подхода, который обеспечивает высокое качество работы модели на широком спектре задач. Особое внимание в процессе работы уделяется качественному отбору и подготовке данных. Этот шаг критически важен, поскольку качество и универсальность итоговой модели напрямую зависят от объема и разнообразия тренировочных данных.

1.1. Методы кодирования текстов

С появлением нейронных сетей в области обработки естественного языка они стали основным методом текстового анализа. Например, нейронная сеть Doc2Vec [24] использует контекстные слова вокруг каждого слова в документе и добавляет уникальный вектор для каждого документа. Модель оптимизируется так, чтобы предсказывать слова из контекста, это позволяет в конечном итоге получить векторное представление для каждого документа, учитывающее его содержание и контекст. Однако, Doc2Vec включает высокие требования к данным, так как для эффективного обучения модели необходимы сотни тысяч документов, каждый из которых должен содержать тысячи слов, что может быть проблематично при дообучении под конкретную задачу [23].

Позднее было предложено использование рекуррентных нейронных сетей, таких как LSTM [16]. Эти модели обрабатывают входной текст рекуррентно, постепенно обновляя вектор эмбеддингов, дополняя его новыми признаками. Такие модели требовали меньше данных и показывали более высокое качество, чем Doc2Vec. Однако из-за последовательной обработки рекуррентные сети сталкиваются с трудностями при работе с длинными документами — обработка каждого следующего токена зависит от предыдущего, что исключает возможность параллельного обучения и использования моде-

ли. В результате аппаратные ресурсы при работе с рекуррентными сетями используются не полностью.

Для решения проблемы была предложена архитектура Transformer [5], обеспечивающая параллельную обработку токенов входной последовательности и эффективное использование ресурсов, что улучшило качество обработки естественного языка. Одной из самых успешных моделей на основе Transformer является BERT (Bidirectional Encoder Representations from Transformers) [6]. BERT, оптимизированный для задач обработки естественного языка, таких как классификация текста, распознавание сущностей и других, значительно улучшил результаты, установив новые стандарты качества и производительности, что делает его подходящим для реализации энкодеров текстов.

1.1.1. Подходы на основе BERT

В модели BERT векторное представление всего текста обычно выделяется двумя способами: среднее по нормализованным эмбеддингам слов или эмбеддинг первого токена – [CLS]. Однако, во время обучения у модели нет прямой задачи сохранить весь смысл текста каким-либо из этих двух способов. Более того, эффективным способом сравнения двух текстов оказалась конкатенация их в один, выделение единого векторного представления и обучение поверх линейного полносвязного слоя в качестве классификатора. При обучении такой модели её трудно переиспользовать для других задач. Например, для поиска текста в корпусе из 10000 документов придётся совершить $\frac{10000 \times 9999}{2}$ кодирований, что является крайне ресурсоёмким процессом.

Авторы Sentence-BERT (SBERT) [40] предложили гораздо более эффективный подход к обучению энкодеров текстов. Вместо того чтобы вычислять один эмбеддинг для двух текстов и на его основе предсказывать общие для текстов признаки, можно вычислить эмбеддинги для каждого из текстов и сравнивать их. Они реализовали сиамскую структуру обучения: две копии модели, делящие веса, параллельно учатся на парах текстов. Итоговые эмбеддинги текстов сравниваются при помощи одной из предложенных авторами мер подобия, зависящих от задачи. Такой подход устраняет существую-

щее до него ограничения. Становится возможным эффективно использовать эмбеддинги для таких задач, как семантическое сходство и кластеризация. Скорость поиска релевантного документа на наборе из 10000 текстов сокращается с 65 часов до примерно 5 секунд. Помимо этого, авторы экспериментировали с триплетами моделей, дополнительно добавляя в обучающую выборку отрицательные пары.

Результаты этих улучшений были значительными. SBERT добился повышения производительности на бенчмарке SentEval. Например, на семи различных задачах Semantic Textual Similarity (STS) SBERT улучшил показатель корреляции Спирмена на 5.5 пунктов по сравнению с Universal Sentence Encoder [54] – предыдущей лучшей моделью.

1.1.2. Функции потерь

В обучении текстовых энкодеров можно использовать различные функций потерь, каждая из которых обладает своими уникальными особенностями и применениями. В статье [54] были предложены сразу три функции потерь, каждая для своей задачи: TripletLoss, MSELoss и CosineSimilarityLoss. Эти функции потерь оптимизируют эмбеддинги, чтобы они лучше отражали семантическую близость текстов. TripletLoss оптимизирует триплеты (текст, положительная и отрицательная пары) так, чтобы положительный пример был ближе к целевому тексту, чем отрицательный, MSELoss минимизирует среднеквадратичное отклонение между эмбеддингами пар, что полезно для задач регрессии, где важна точность предсказаний. CosineSimilarityLoss минимизирует косинусное расстояние между эмбеддингами положительных пар и максимизирует его для отрицательных пар.

Одной из ключевых функций потерь в задачах контрастивного обучения и обучения текстовых энкодеров является InfoNCE (Information Noise-Contrastive Estimation) [26]. Основной принцип работы этой функции заключается в максимизации различий между положительными и отрицательными примерами в рамках одной задачи.

Для заданной пары текстов функция InfoNCE определяет положительный пример как релевантную пару, а отрицательные примеры выбираются

из других текстов в батче. Функция потерь рассчитывается по следующей формуле:

$$\mathcal{L}_{\text{cont}} = -\frac{1}{n}\sum_{i}\log\frac{\exp(s_\theta(q_i, p_i))}{\exp(s_\theta(q_i, p_i)) + \sum_j \exp(s_\theta(q_i, p_{ij}^-))} \tag{1}$$

,

где $s_\theta(q, p)$ – это функция оценки, параметризованная θ, которая вычисляет сходство между *query* и *passage*. В данной формуле p_{ij}^- представляет отрицательные примеры для i-й пары, а n – количество положительных примеров в батче.

InfoNCE использует механизм in-batch negatives, при котором отрицательные примеры выбираются из других пар в текущем батче, что позволяет эффективно использовать вычислительные ресурсы и улучшает стабильность обучения. В процессе обучения задача функции потерь заключается в том, чтобы максимизировать сходство положительных примеров и минимизировать сходство отрицательных примеров, тем самым улучшая качество эмбеддингов.

InfoNCE особенно полезна в задачах, где требуется обучение на больших объемах неразмеченных данных, таких как задачи извлечения информации, кластеризации и классификации текстов. Контрастивное обучение с использованием данной функции потерь позволяет моделям лучше различать семантически схожие и различающиеся тексты, что критично для успешного выполнения задач оценки семантической близости и других задач, связанных с эмбеддингами.

Однако, так как в качестве функции оценки близости в InfoNCE используется косинусное расстояние, векторы, для которых это значение близко к единице или, наоборот, к нулю, могут испытывать затухание градиентов из-за свойств косинусной близости. Статья, посвященная функции потерь AnglE [26], акцентирует внимание на повышении эффективности обучения модели для задач оценки семантической близости текстов.

Основная проблема существующих моделей эмбеддингов текста заключается в затухании градиентов, вызванном насыщенными зонами косинусной функции, используемой в оптимизационных задачах. Для решения этой

проблемы в статье предлагается новая модель эмбеддингов текста, оптимизированная по углам в комплексном пространстве, называемая AnglE. Этот подход позволяет избежать затухания градиентов и улучшает точность оценки семантической близости текстов, обеспечивая более стабильное и эффективное обучение модели.

Главная идея AnglE заключается во введении оптимизации углов в комплексном пространстве для эффективного уменьшения негативного воздействия насыщенных зон косинусной функции, что может препятствовать обучению модели. AnglE оптимизирует не только косинусное сходство между текстами, но и угол между эмбеддингами в комплексном пространстве. Это позволяет модели лучше различать тексты, особенно в сложных случаях, когда градиенты косинусной функции стремятся к нулю. Эксперименты показали, что модели, обученные при помощи AnglE, превосходят существующие в метрике STS. В различных симметричных задачах, включая короткие и длинные тексты, AnglE демонстрирует высокое качество эмбеддингов текста. Поэтому эта функция потерь является наиболее перспективной для обучения семантических энкодеров текстов.

1.1.3. Инструкции

Инструкция – это префикс (иногда суффикс), который добавляется к тексту и повышает дискриминативную способность энкодера текстов. Модели становится гораздо проще понять, хотим ли мы найти релевантный текст, ответ на вопрос из медицины или перефразированное предложение. Это происходит благодаря тому, что мы буквально сообщаем модели, эмбеддинги каких текстовых данных мы хотим получить, к какой теме относящихся и для решения какой задачи.

Первая работа [2], в которой было предложено использовать инструкции, была направлена на дообучение больших языковых моделей. Позднее в работе Su и соавторов [36] было предложено использовать инструкции уже для обучения энкодеров текстов. Для каждого текста из пары они указывают тип данных для кодирования, тип данных для поиска и домен данных. Пример таких инструкций:

Это заголовок для поиска статей из источника Газета: {заголовок}
Это статья для поиска заголовка из источника Газета: {статья}

Хоть такие инструкции достаточно эффективны на задачах, смежных с тренировочной выборкой, однако на этапе валидации при появлении новой эвристики или специфических данных из ресурса, о котором модель ничего не знает, эти инструкции перестают подходить. Так, в качестве новых инструкций разработчикам приходится использовать те, что модель ещё не знает, либо вовсе опустить их. В обоих случаях модель сильно теряет в качестве.

Кроме того, не ясно какую новую инструкцию следует выбрать для задачи выделения признаков, ведь эту задачу невозможно добавить на этапе обучения. В работах [21, 35] с этой проблемой справились, используя следующий дизайн инструкций:

Инструкция: {постановка задачи} query: {запрос}
Инструкция: {постановка задачи} passage: {текст}

В данном контексте под запросом (*query*) и текстом (*passage*) понимается запрос пользователя к базе данных с целью поиска релевантных текстов. Эта концепция позволяет использовать соответствующие инструкции в задачах выделения признаков, сохраняя «*query*: » и разделяя по типу инструкции на основе используемых запросов и текстов в зависимости от типа данных. Такой подход позволяет избежать дополнительной фильтрации данных и разделения крупных корпусов текстов на кластеры, предотвращая появление перекрестных доменов.

Также, к упрощению пришли авторы моделей BGE [8]. Они используют инструкцию лишь для одного текста из пары – целевого в поиске. Этим они помогают модели разделить запрос и документ. Пример инструкции, которую они используют:

search relevant passages for the query {запрос}

Так как их модель направлена исключительно на задачу извлечения релевантных текстов, нет данных о том, как эта инструкция будет работать в задачах извлечения признаков.

Таким образом, существующие подходы к написанию инструкций нуждаются в дополнительных шагах адаптации для повышения универсальности итоговой модели.

1.1.4. Дообучение под конкретную задачу

Часто оказывается необходимым дообучение энкодера текстов под новую для него задачу, при этом качество на других задачах может заметно ухудшиться. Это хорошо известная в области машинного обучения проблема катастрофического забывания, в связи с которой модели начинают хуже справляться с рядом задач после дополнительного обучения. Существуют два традиционных подхода для решения данной проблемы. Первый подход предполагает смешивание исходных данных с данными, использованными в процессе дообучения. Второй метод заключается в объединении весов модели до и после дообучения. Под объединением здесь подразумевается сложение весов с некоторыми коэффициентами.

В работе [27] авторы предложили оптимальный способ объединения весов моделей на основе нескольких примеров пар текстов. Так, коэффициэнт для i-той модели в списке подбирается по Формуле 2. В этой формуле $\mathcal{L}(\mathcal{M}_i, E)$ – это значение функции потерь для i-той модели на каждой из переданных пар из E, τ – температура. Чем больше значение $\mathcal{L}$, тем меньший коэффициент присваивается модели.

$$w_i \leftarrow \text{softmax}(-\mathcal{L}(\mathcal{M}_i, E_t)/\tau) \tag{2}$$

1.2. Обзор датасетов для обучения энкодеров текстов

Данные для обучения энкодеров текстов представляют собой пары текстов, связанные различными эвристиками. В контексте данных для обучения энкодера текстов, под эвристикой понимается определённая семантическая связь между парой текстов. Эвристики могут подразделяться на симметричные и асимметричные. Согласно исследованию [36], к симметричным эвристикам относятся те, в которых «оба текста имеют одинаковую форму и семантический смысл». Форма здесь строго определяется как длина тек-

ста, измеряемая количеством лингвистических единиц: слово, предложение, абзац или целый текст. Примеры симметричных эвристик: (текст, логическое следствие), (текст перефразированный текст), (текст, перевод). Асимметричными эвристиками являются все остальные, например, эвристики: (вопрос, ответ), (запрос, релевантный документ), (заголовок, статья).

Пары текстов могут быть как положительными, так и отрицательными. Положительные пары текстов близки по смыслу, хотя могут существенно отличаться по форме и стилю написания. Такие тексты, несмотря на различия в формулировках и лексическом составе, передают схожую информацию и имеют одинаковую тематику. С другой стороны, отрицательные пары текстов зачастую похожи по написанию и могут касаться одной и той же темы, однако содержат противоположные смыслы. Эти тексты могут использовать похожие слова и фразы, но выражать разные, а иногда и противоположные точки зрения или факты. Такие различия особенно важны для задач обработки естественного языка, где требуется точное понимание семантического содержания текстов. Большинство датасетов не содержат такие пары по умолчанию, поэтому были предложены различные методы по их поиску внутри корпуса.

1.2.1. Поиск отрицательных пар

Сложные отрицательные пары (hard negatives, HN) близки к положительным (positives, P) для конкретного текста и создают трудности для модели в процессе обучения. Вероятность случайного включения такой пары в батч размером b из общего корпуса $|C|$ стремится к нулю, что отражено в Формуле 3. Это говорит о необходимости искусственного создания сложных отрицательных пар.

$$p = \frac{b|HN|}{|C|^2} \sim 0 \tag{3}$$

Исследование, проведенное Chen и соавторами [7], демонстрирует использование батчей размером до 67200 при общем числе пар более миллиарда, что подчёркивает масштабность необходимых вычислительных мощностей.

Рассмотрим существующие методы поиска отрицательных пар текстов

[4]. 1) Локальные отрицательные пары в мини-батчах выбираются из текущего батча. Этот метод подходит для обучения моделей представлений слов и изображений, но его эффективность снижается в семантическом кодировании из-за доминирования неинформативных отрицательных примеров, что замедляет сходимость обучения. 2) Использование алгоритма ранжирования BM25. В этом варианте поиска отрицательных пар они выбираются из топа документов, полученных с помощью BM25, из-за чего модель может перенаправлять фокус исключительно на них. 3) Выбор случайных отрицательных пар из корпуса документов. Проблемой данной стратегии является частый выбор слишком лёгких, неинформативных примеров для модели. 4) Комбинация методов BM25 и выбора случайных отрицательных пар. Данный подход позволяет модели получать разнообразные отрицательные примеры, однако это все равно может привести к доминированию неинформативных примеров. 5) Метод ANCE (Approximate Nearest Neighbor Negative Contrastive Learning), использующий асинхронно обновляемый индекс «примерного ближайшего соседа» (approximate nearest neighbour, ANN) для выбора глобальных отрицательных примеров из всего корпуса документов. По сравнению с выше описанными подходами к поиску отрицательных пар текстов данный метод решает проблему неинформативных локальных отрицательных примеров [4] и оказывается наиболее эффективным, так как обеспечивает выбор самых сложных отрицательных примеров для текущей модели, улучшая сходимость обучения. В ANCE отрицательные примеры выбираются из топ-документов, полученных энкодером текстов, а также используется асинхронное обновление индекса через каждые несколько минибатчей или раз в эпоху для поддержания актуальности отрицательных примеров в ходе обучения модели. Эксперименты показали, что ANCE значительно улучшает точность энкодеров текстов по сравнению с другими методами выбора отрицательных примеров. В задачах веб-поиска,вопросно-ответных систем и в коммерческих поисковых системах ANCE достиг результатов, сравнимых с каскадными моделями на основе BERT, при этом обеспечивая в 100 раз большую эффективность [4].

1.2.2. Данные из открытых источников

На данный момент, к сожалению, количество доступных датасетов для русского языка значительно меньше, чем для английского или китайского. Например, в статье [36] и при создании моделей all-mpnet-base-v2[2] и bge-m3 [7], для обучения использовалось более миллиарда пар.

В настоящее время существуют платформы, предоставляющие удобный поиск открытых датасетов, такие как HuggingFace и Kaggle. Эти ресурсы облегчают процесс сбора данных за счет различных фильтров по задачам и языкам. Несмотря на доступность, данные могут иметь различный формат, дальнейшее их использование требует унификации.

1.3. Методы валидации моделей

Валидация является критически важным этапом в машинном обучении. Правильный и разносторонний выбор метрик оценки помогает понять сильные стороны модели и выявить её недостатки. Для обеспечения надежности тестирования требуется исключение пересечения между тренировочными и тестовыми выборками. Учитывая, что данные в рамках одного датасета зачастую похожи, зачастую исключаются из даже тренировочные части, для оценки генерализации модели.

1.3.1. Рейтинг русскоязычных энкодеров предложений

Бенчмарк «Рейтинг русскоязычных энкодеров предложений» или «Энкодечка» [13] является наиболее актуальным инструментом для оценки энкодеров текстов, ориентированных на русский язык. Он включает восемь метрик, направленных на извлечение смысла всего текста, а также две метрики NER (Named Entity Recognition), не относящиеся к семантическому кодированию всего текста, поэтому они не в этой работе не рассматриваются. Кратко разберём задачи из «Энкодечки»:

[2]huggingface.co/sentence-transformers/all-mpnet-base-v2

Semantic Text Similarity (STS): Задача оценки степени семантического сходства между текстами. Качество определяется с помощью корреляции Спирмена между косинусной близостью эмбеддингов и оценками сходства, выставленными людьми.

Paraphrase Identification (PI): Определение, являются ли два текста парафразами. Метрика оценки качества — корреляция Спирмена косинусной близости эмбеддингов с оценками сходства.

Natural Language Inference (NLI): Задача установления логической взаимосвязи между парами предложений (entailment, contradiction, neutral), на основе XNLI. Оценка точности производится с помощью трёхклассовой логистической регрессии, обученной на косинусной близости эмбеддингов предложений.

Sentiment Analysis (SA): Задача классификации предложений по их эмоциональной окраске. Метрика оценки качества — точность логистической регрессии или KNN.

Toxicity Identification (TI): Задача определения токсичности комментариев. Качество определяется с помощью ROC AUC логистической регрессии или KNN.

Inappropriateness Identification (II): Определение неподобающих комментариев. Оценка качества производится на основе ROC AUC логистической регрессии или KNN.

Intent Classification (IC) и Cross Intent Classification (ICX): Определение намерений пользователя. Метрика качества — точность логистической регрессии или KNN. В задаче IC используется русскоязычный корпус, в ICX — кросс-язычная версия, где обучение проводится на английском корпусе, а тестирование — на русском.

Все вышеуказанные задачи можно разделить на три категории: оценка схожести текстов (STS, PI, NLI), бинарная классификация (TI, II) и многоклассовая классификация (SA, IC, ICX). Стоит отметить, что метрика ICX выделяется, так как мультиязычные модели на ней будут значительно превосходить моноязычные.

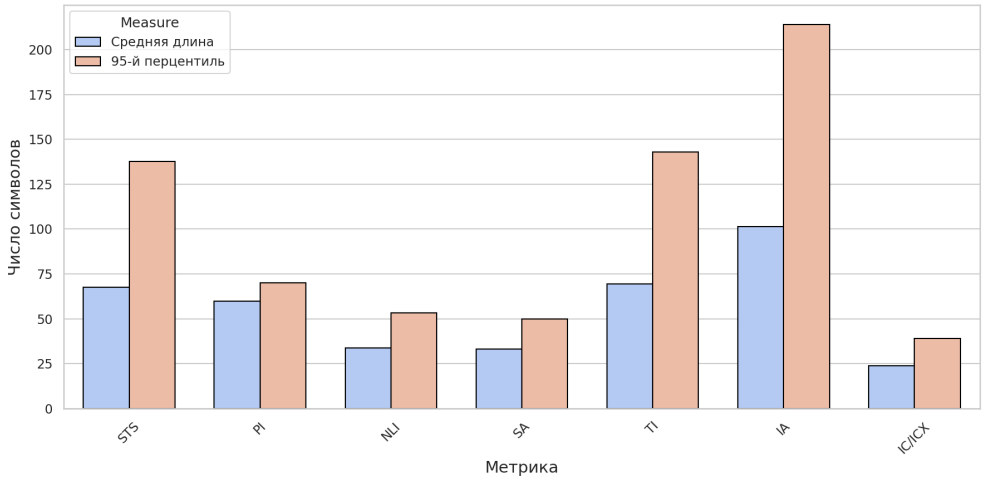


Рис. 1: Средние длины и 95-й перцентиль длины текстов из «Энкодечки»

Бенчмарка «Энкодечка» является наиболее актуальным и популярным для русского языка на текущий момент. Но учитывая актуальность и специфику RAG-задач, становятся очевидными некоторые недостатки «Энкодечки», поскольку в нём отсутствуют метрики, коррелирующие с задачами извлечения информации. Во-первых, все задачи в «Энкодечке» симметричные, тогда как RAG-задачи асимметричные. Во-вторых, RAG часто предполагает работу с очень длинными текстами. Анализ длины текстов в задачах из «Энкодечки» показывает, что 95-й перцентиль распределения длины текстов не превышает 250 символов, что изображено на Рисунке 1. При переводе в токены средняя длина последовательностей сокращается ещё больше. В то же время, максимальная длина входной последовательности, поддерживаемая большинством современных энкодеров, а также разработанным мной, равна 512 токенам. Поэтому необходимо дополнять валидацию задачами извлечения информации, содержащими длинные тексты.

1.3.2. Задачи извлечения релевантной информации

Первой и наиболее масштабной задачей извлечения релевантной информации, является MIRACL (Multilingual
Information Retrieval Across a Continuum of Languages) [32] – датасет который разработан для соревнования WSDM 2023 и фокусируется на поиске документов по запросам пользователей. Он содержит 18 языков, но структурирован для поддержки моноязычных поисковых задач, что позоляет использовать только его русскую часть.

Каждый язык в датасете имеет набор запросов и связанных с ними документов. Суждения о релевантности того или иного документа конкретному запросу выносились носителями языка, что обеспечивает высокое качество данных для обучения и оценки. Все документы выбраны из Википедии. Формат датасета поддерживает использование метрик оценки информационного поиска: Recall и nDCG, формулы вычисления которых я указал в Приложении 3.

Стоит упомянуть и датасет MLDR (Multi Long Doc Retrieval) [7], который содержит особо длинные тексты, длина каждого из которых превышает 3000 символов. Создатели датасета на основе документов из Википедии и ещё двух корпусов просили большую языковую модель сгенерировать вопросы. Вопрос и выбранный документ составляют новую пару текстов для датасета. Авторы предложили использовать его в качестве задачи извлечения релевантной информации в связке с MIRACL, для более объективной картины работы моделей.

1.3.3. MTEB

Наиболее объёмный на данный момент мультиязычный бенчмарк MTEB (Massive Text Embedding Benchmark), основным языком которого всё же является английский, содержит 8 задач, содержащих подзадачи на русском языке, которые можно разделить на 4 категории:

- STS22, STSBenchmarkMultilingualSTS, XNLI – оценка похожести текстов;

- MassiveScenarioClassification, MassiveIntentClassification, OpusparcusPC – классификация;
- XQuADRetrieval – извлечение релевантных текстов;
- MLSUMClusteringP2P, MLSUMClusteringS2S – кластеризация.

1.4. Выводы

Методы обучения семантических энкодеров мало изменились с появления сиамской архитектуры [40]. Больше работы проводится над выбором данных для обучения: созданием новых датасетов, повышением качества данных, улучшением дискриминативности при помощи инструкций.

Несмотря на значительные достижения в области кодирования текстов, остаются вызовы, такие как необходимость в больших объемах данных для обучения, адаптация моделей к новым типам данных и доменам. Эти вызовы подчеркивают важность непрерывных исследований и разработок в данной области.

Продолжающееся развитие методов кодирования текстов обещает еще больше улучшить качество сохранение семантического смысла, что будет способствовать более глубокому пониманию и более точному интерпретации текстов в самых различных областях применения.

2. Разработка алгоритма обучения модели

Из обзора предметной области становится ясно, что список алгоритмов для обучения энкодеров текстов ограничен. Большинство исследований фокусируются на увеличении объема данных, что особенно эффективно для мультиязычных или англоязычных моделей, где число положительных пар может достигать миллиардов. Однако при недостатке данных, что характерно для русского языка, необходим более качественный и тщательный подход к обучению. Поэтому в данной работе я постарался воспроизвести наиболее актуальные подходы, при этом добавляя собственные улучшения.

2.1. Выбор методов

Работа [7] показала, что качественное предобучение под задачу кодирования текстов крайне сильно влияет на итоговый результат. Однако этот этап требует больших вычислительных ресурсов и значительных объёмов данных, которых у меня не было. Поэтому, в качестве предобученной модели я использовал deepvk/deberta-v1-base[3]. Выбор обусловлен оценкой на бенчмарке RussianSuperGlue [45], результаты метрик указаны в карточке модели.

В работе C-PACK [8] был предложен двухуровневый этап дообучения. Цель первого «unlabeled» этапа состоит в грубой настройке модели на менее качественных данных, чем на итоговом этапе, но всё равно способных передать определённую информацию. На этом этапе я обучил два варианта модели. Первая модель была обучена на русскоязычной части корпуса mMarco [63], а вторая — на mMarco и парах последовательных текстов статей из Википедии. Результаты «unlabeled» дообучения приведены в разделе 4.2. Первый вариант показал больший прирост на целевой метрике, поэтому в дальнейшем использовал его.

Из обзора собранных данных видно, что отрицательных пар значительно меньше, чем положительных. Поэтому я использовал упрощённый подход ANCE, аналогичный тому, что применялся при обучении модели BGE [8].

[3]https://huggingface.co/deepvk/deberta-v1-base

Упрощение заключалось в следующем: для поиска отрицательных пар использовалась внешняя модель, которая хорошо справлялась с задачей кодирования. Я выбрал модель baai/bge-m3[4] как наиболее точную. Кроме того, такие пары не были пересчитаны во время обучения, а фиксировались на нулевом шаге. Для каждого текста, который не содержал отрицательной пары, я выбирал её внутри корпуса. Выбор осуществлялся случайно из диапазона 10-210 ближайших эмбеддингов к целевому тексту.

В своей работе я стремился разработать модель, которая эффективно работает с различными видами и доменами данных, не фокусируясь на тренировочной выборке. Поэтому я предложил собственный дизайн инструкций, упростив подход, представленный в работе [35]. Так, были сохранены только части «query: » и «passage: », чтобы разделять симметричные и асимметричные данные. Метку «query: » я конкатенировал с каждым текстом для симметричных данных, а для асимметричных данных использовал «query: » и «passage: » для обозначения запросов и текстов соответственно. Пример такой инструкции представлен на Рисунке 2.

Рис. 2: Пример инструкции, которую, добавляемой к асимметричной эвристике (вопрос, ответ).

В начале экспериментов я проводил второй «labeled» этап дообучения на всех собранных данных. Однако, функция потерь на валидационной части асимметричных данных не снижалась. Более того, оценка модели по целевым метрикам показала значительное ухудшение качества на задачах выделения релевантных текстов. Проблема заключалась в значительном различии между асимметричными и симметричными данными, и использование простых инструкций лишь незначительно помогало в её решении.

[4]BAAI/bge-m3

Основываясь на подходе, предложенном в работе [27], я предложил дизайн с разделением данных по симметричности и обучением двух различных моделей — каждой для своей задачи, с соответствующими инструкциями. Кроме того, стало возможным усложнить функцию потерь с помощью AnglE, которая работает исключительно для симметричных задач. Она вычисляет близость в комплексом пространстве, нивелируя проблемы косинуса. Поэтому, я суммирую InfoNCE и AnglE, с коэффициэнтами w_1, w_2, пример указан на Формуле 4.

$$L_f = w_1 \cdot L_{InfoNCE} + w_2 \cdot L_{AnglE} \tag{4}$$

Затем, используя LM-Cocktail, я складываю веса с подобранными методом коэффициэнтами на основе нескольких примеров их валидационной выборки и получаю итоговую модель.

2.2. Подготовка данных для обучения

В машинном обучении итоговая модель в значительной степени определяется качеством и объемом данных, используемых для её обучения. Я посчитал, что рассмотренных в разделе 1.2.2 данных будет недостаточно для обучения качественной модели. Во-первых, большая их часть является переводами с английского. Во-вторых, они содержат относительно мало отрицательных пар. В связи с этим, было предложено создать два собственных датасета.

2.2.1. RuHNP

Как отмечено в исследовании [17] и формально обосновано в разделе 1.2.1, для эффективного обучения текстового энкодера крайне важно использовать сложные отрицательные пары. Сложная отрицательная пара – два тесно связанных текста, однако передающие различные смыслы. В Таблице 1 приведены существующие датасеты перефразированных текстов на русском языке, доступные в открытых источниках. Из них только пять небольших датасетов содержат отрицательные пары. При этом, лишь в ра-

боте RuPAWS [44] были предложены несколько действительно сложных для моделей трансформаций текстов, таких как замена слов и взаимный обмен имён собственных. Однако в этой же статье все итоговые модели продемонстрировали ухудшение качества на основной метрике, что может говорить о низкой генерализуемости датасета.

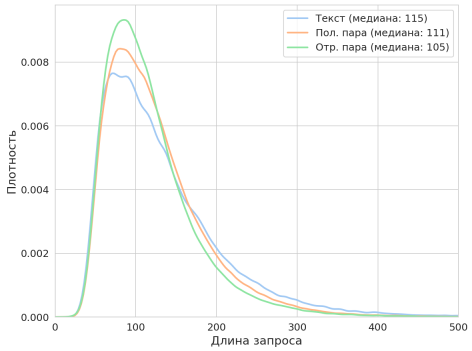
Таблица 1: Обзор датасетов перефразированных текстов на русском языке. *Число положительных пар.

Название	**Размер***	**Перевод**	**Синтетика**	**Отр.**
RuParaphraser [38]	7.2k	-	-	+
StsbMultiMt [29]	8.6k	+	-	+
Paraphrases [22]	18k	-	-	+
RuFacts [59]	4.7k	-	-	+
RuPAWS [44]	7.5k	-	+	+
Tapaco [49]	250k	-	-	-
RudetoxifierDataDetox [30]	31k	-	+	-
RuParadetox [39]	11k	-	-	-
Ru-paraphrase-NMT [12]	1M	+	-	-
RuHNP (наша работа)	500k	-	+	+

Основываясь на этих наблюдениях, я собрал RuHNP (Russian Hard Non-Paraphrases) – собственный датасет перефразированных текстов, уделяя особое внимание объему и качеству отрицательных пар. Для каждого из более чем ста тысяч тексов из Википедии, я генерирую с помощью ChatGPT-3.5-turbo по пять положительных и отрицательных пар. Промпт генерации был написан на основе рекомендаций OpenAI и находится в Приложении 2.

После первичной генерации датасета, из полученных 103968 ответов модели были отфильтрованы тексты, не соответствующие заданным критериям: короткие ответы (меньше 6 символов), данные, нарушающие структуру JSON и ответы, не содержащие ровно 5 положительных и 5 отрицательных пар. В результате этой фильтрации для дальнейшего использования осталось 102590 примеров. Они были разделены на тренировочную, валидационную и тестовую выборки в соотношении 100000/590/2000 соответственно. Каждый из итоговых примеров содержит текст, 5 положительных и 5 отрицательных пар. То есть тренировочная выборка содержит полмиллиона

положительных пар. Датасет представлен в этом формате и разделён на данном этапе, чтобы исключить пересечение тренировочной выборки с валидационной и тестовой. Распределения длин текстов и медианные значения приведены на рисунке 3а.



(a)

query: После войны продолжал службу в Советской армии на штабной работе.

pos: На штабе в Советской армии продолжал служить после войны.

neg: После войны переквалифицировался и ушел из Советской армии на завод.

(b)

Рис. 3: (a) распределение длин исходных текстов и сгенерированных пар; (b) один пример из полученного датасета.

Для анализа качества полученного датасета мной была предложена следующая методика. Во-первых, выбираются несколько обученных моделей, специализирующихся на задачах определения перефразированных текстов и измерения семантической близости для русского языка. Во-вторых, проводится дообучение выбранных моделей на тренировочном наборе данных RuHNP, используя единый пайплайн обучения: максимальная длина последовательности токенов равна 64, длительность обучения составляет 1 эпоху, коэффициент скорости обучения настраивается в зависимости от размера модели. В-третьих, используются валидационные части датасетов, содержащих отрицательные пары из Таблицы 1. Далее полученные результаты усредняются по всем использованным моделям и строятся графики распределения косинусных близостей до и после процесса обучения.

Модели, выбранные для эксперимента, представлены в Таблице 2. Модели 1, 3, 5, 6, 7 присутствуют в лидерборде «Энкодечка» [13] и демонстрируют высокое качество на релевантных для задачи метриках (STS, PI, NLI). Результаты эксперимента представлены на Рисунке 4. Заметим, что для всех датасетов, кроме RuPAWS, распределение косинусных близостей

Таблица 2: Набор моделей для проведения эксперимента по оценке RuHNP

№	Модель	lr
1	symanto/sn-xlm-roberta-base-nli [34]	2×10^{-6}
2	cointegrated/rubert-base-paraphrase-detection [14]	2×10^{-6}
3	intfloat/multilingual-e5-large [35]	6×10^{-7}
4	intfloat/multilingual-e5-base [35]	2×10^{-6}
5	sentence-transformers/mult-mpnet-base-v2 [40]	2×10^{-6}
6	sentence-transformers/LaBSE	2×10^{-6}
7	BAAI/bge-m3 [7]	6×10^{-7}

для отрицательных пар существенно сдвинулось влево, отдалившись от распределения для положительных пар. Также после дообучения модели стали значительно лучше справляться с разделением пар из валидационной части датасета RuHNP, что свидетельствует о его консистентности. Увеличение дискриминативности на других датасетах указывает на хорошую генерализуемость RuHNP. Даже после короткого дообучения без серьёзного подбора гиперпараметров улучшилась способнось моделей определять семантически отличающиеся тексты.

2.2.2. RuWANLI

NLI (Natural Language Inference) – представляет собой задачу определения логической связи между двумя текстами. На основе текста A (предпосылки) модели необходимо определить класс примера, установив отношение текста B (гипотеза) тексту A: является ли он его логическим следствием, противоречием или вовсе не относится к тексту A.

На данный момент все относительно большие NLI-датасеты на русском языке являются переводами с английского. Из-за этого в тексте содержится немало артефактов и ошибок перевода, что усложняет процесс обучения. Поэтому я решил собрать новый NLI-датасет, следуя подходу WANLI [55], так как он включает этап фильтрации с помощью асессоров, что значительно улучшает итоговое качество датасета. В Таблице 3 представлено сравнение NLI-датасетов на русском языке и моего датасета. В столбцах указано, является ли датасет переведённым или сгенерированным.

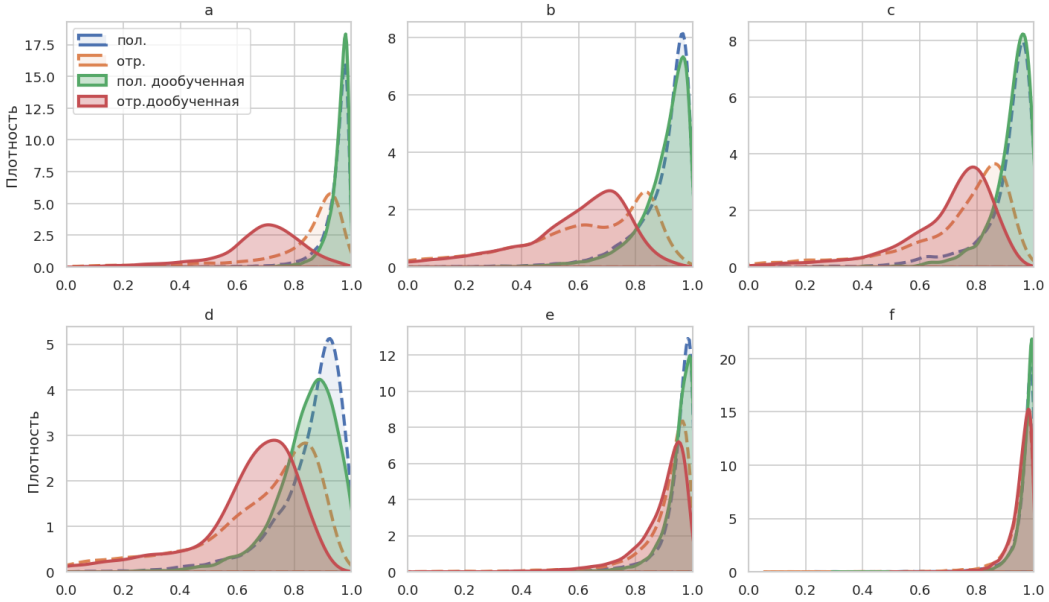


Рис. 4: Распределения для косинусных близостей валидационных частей датасетов: (a) RuHNP; (b) paraphrases [22]; (c) ru paraphraser [38]; (d) stsb multi [29]; (e) ru facts [59]; (f) RuPAWS [44].

RuWANLI (Russian-Worker-AI Collaboration for NLI) – синтетический русскоязычный NLI датасет, для создания которого я воспроизвёл алгоритм WANLI [55]. Процесс начинается с автоматической сборки сложных для обучения примеров из базового датасета AllNLI. Отбор таких примеров производится на основе анализа карт данных [15]. Для всех отобранных примеров модель deepvk/deberta-v1-base вычисляет эмбеддинги. С помощью них ищутся четыре наиболее близких соседа по косинусной близости в рамках того же класса. Эта процедура позволяет сформировать группы примеров, на основе которых далее строится контекст для создания новых данных. Генерация новых примеров осуществляется с помощью модели ChatGPT-3.5-turbo, на основе одного контекста создаются пять новых предложений. Промпт генерации указан в Приложении 1.

Ключевым аспектом разработки этого датасета является фильтрация сгенерированных данных. Для этого я привлек человеческих аннотаторов для проведения ручной разметки на платформе «ВКонтакте». Всего в оценке датасета приняло участие 119 человек, для каждого из 74258 текстов бы-

Таблица 3: Существующие NLI датасеты на русском языке. *Содержит переводы add_one_rte, anli, copa, fever, help, iie, imppres, joci, mnli, monli, mpe, qnli, scitail, sick, snli. **Содержит переводы anli, fever, ling, mnli, wanli.

Название	**Размер**	**Перевод**	**Синтетика**
SNLI [62]	550k	+	-
MNLI [56]	393k	+	-
XNLI [58]	400k	+	-
ANLI [3]	161k	+	-
WANLI [55]	103k	+	-
MedNLI [43]	25k	+	-
RCB [45]	1.1k	-	-
Terra [45]	6.1k	-	-
nli-rus-translated-v2021* [11]	1.7M	+	-
multilingual-NLI-26lang-2mil7** [25]	125k	+	-
RuWANLI (наша работа)	100k	-	+

ло выставлено от 5 до 7 оценок. Дополнительно к одному из трёх классов, аннотаторы могли маркировать данные как «плохой пример». Эта метка является сигналом для возможного исключения данных из финальной версии датасета. Процесс исключения происходил, если большинство аннотаторов проголосовали за плохое качество конкретного примера. Кроме того, отдельно рассматривалась альтернатива с исключением при наличии хотя бы одной метки «плохой пример», но в этом случае было удалено слишком много данных, поэтому пришлось отказаться от такой стратегии.

На Рисунке 5 показаны графики зависимости числа удаляемых данных от количества исключённых аннотаторов для двух стратегий. Первая стратегия предполагала удаление примера при наличии хотя бы одной метки «плохой пример», а вторая – при наличии большинства. Кроме того удалялись двусмысленные примеры, то есть те, для которых не удалось выбрать однозначное большинство голосов.

Несмотря на то, что модель должна генерировать текст в соответствии с определённым классом, дополнительная верификация аннотаторами показала, что качество сгенерированных примеров не всегда соответствует ожиданиям: лишь около половины примеров получили одинаковую оценку от модели и людей.

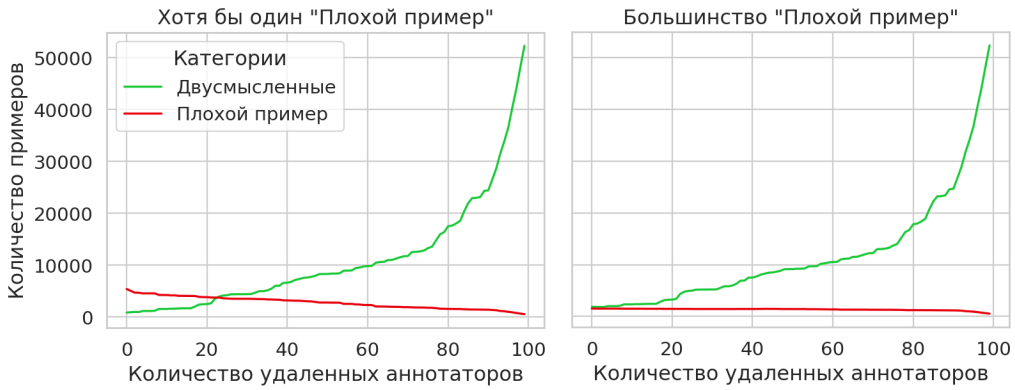


Рис. 5: Графики сравнения количества удаляемых данных при двух стратегиях. «Двусмысленные» образцы это те, для которых нет однозначного большинства за какую-либо категорию.

В процессе подготовки данных одной из ключевых проблем оказалось отсутствие доверенных аннотаторов (gold labels), что усложнило задачу фильтрации аннотированных результатов. Моя цель заключались в максимизации качества данных и минимизации их потери при очистке датасета от ненадёжных аннотаций. Для этого была разработана методика оценки работы аннотаторов, основанная на степени их несогласия с большинством. Так, для каждого аннотатора было рассчитано, как часто его оценки расходятся с оценками большинства. Далее оценки были отсортированы по возрастанию рассчитанного значения согласованности. Затем, постепенно удалялось по одному «худшему» аннотатору и оценивался индекс Флайса на текущем шаге. Индекс Флайса (Fleiss' kappa) – это статистический показатель, оценивающий согласованность между несколькими аннотаторами при классификации объектов по категориям, с учётом случайного совпадения оценок. Значения индекса варьируются от -1 до 1: 1 означает идеальную согласованность, 0 указывает на случайную согласованность, а отрицательные значения свидетельствуют о систематическом несогласии.

Зависимость индекса Флайса (индекс согласованности между аннотаторами) и процента потери данных от количества удалённых «худших» аннотаторов, то есть тех, которые наименее часто соглашаются с оценками остальных изображена на Рисунке 6. n-ная точка по горизонтали (так как значение fk возрастает) показывает значения индекса Флайса (fk) и процента

потерь данных (dl) после удаления n «худших» аннотаторов.

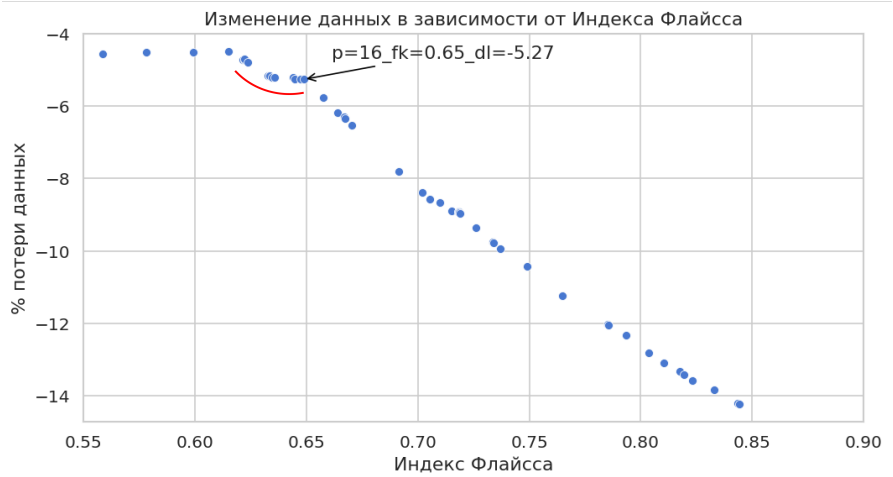
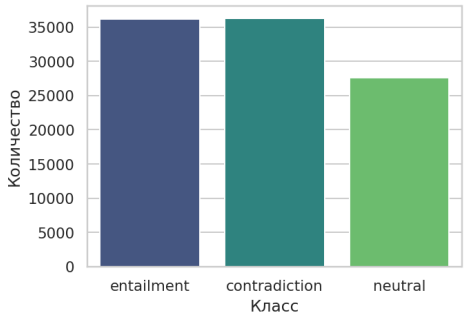


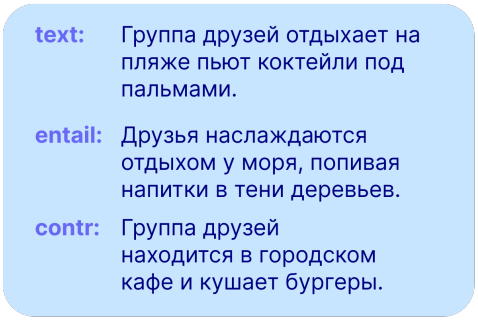
Рис. 6: График зависимости индекса Флайса и процента потерянных данных от количества удалённых «худших» аннотаторов. Стрелкой указано выбранное мной значение.

По результатам анализа графика было решено остановиться на удалении 16 аннотаторов, так как этот выбор соответствует концу «плато» – точке, после которой дальнейшее удаление аннотаторов приводит к существенной потере данных. В этой точке значения индекса Флайса и процента потерь данных составили 0.65 и -5.27% соответственно.

После шага фильтрации размер данных составил 70348 примеров, из которых я отделил 5000 для тестовой выборки. Чтобы полученный набор данных можно было более эффективно использовать при обучении энкодера текстов, необходимо наличие троек (text, entailment, contradiction). Следуя этому, я дополнил алгоритм шагом генерации третьего значения для существующих пар. Аналогично 2.2.1 было предложено использовать ChatGPT, но теперь версию ChatGPT-4. Промпт генерации указан в Приложении 1. В итоге было получено разделение на train/val/test в соотношении 100000/2360/5000. Посмотрим на распределение классов (Рисунок 7a) и на два примера: text/entailment и text/contradiction итогового датасета (Рисунок 7b). Благодаря дополнительному шагу генерации датасет содержит одинаковое число логических следствий и противоречий.



(a) Распределение классов



(b) Пример из данных

Рис. 7: Распределение длин исходных текстов и сгенерированных пар

После создания датасета было решено оценить его качество, обучив модель сначала на существующих NLI датасетах, а затем на их объединении с нашими данными. Для SNLI, MNLI, ANLI я создал комбинированные датасеты, заменяя 65000 примеров оригинального датасета на RuWANLI. В Таблице 4 представлены результаты обучения моделей. Для более чем половины метрик качество на тестовой выборке возросло, но не удалось добиться такой же генерализации, как в оригинальной работе. Вероятно, проблема в том, что полученная синтетика создаётся на основе переводов, а не оригинальных текстов.

Таблица 4: Сравнение качества моделей на тестовой выборке. Зелёным выделены значения, которые выросли при замене части исходного датасета на RuWANLI. Серым указаны значения, не учитывающиеся при подсчёте среднего.

	Тестовая выборка									
	SNLI	**MNLI**	**ANLI**	**XNLI**	**WANLI**	**MedNLI**	**RCB**	**Terra**	**RuWANLI**	**Среднее**
SNLI	87.3	68.8	30.6	67.4	45.4	56.1	23.5	39.6	62	47.3
MNLI	70.9	79.5	29.7	76.5	51.2	57.6	29.4	39.1	66.5	50.6
ANLI	56.3	64.2	45.4	63.7	46.5	54.8	31.8	38.8	62.1	50.8
SNLI ⋄ **RuWANLI**	87.1	68.6	33.1	67.4	46.3	55.9	25.6	38.8	78.4	48
MNLI ⋄ **RuWANLI**	67.8	79.2	31.9	75.8	51.7	58.3	30.5	40.8	79.2	51
ANLI ⋄ **RuWANLI**	58	68.8	43.9	63	45.1	55.2	28.8	38	77.5	50.3

2.3. Результаты подготовки данных

Выберем датасеты из открытых источников и объединим с 2.2.1,2.2.2. Подсчитаем итоговый размер тренировочный выборки (число положитель-

Таблица 5: Итоговый набор датасетов; отмеченные зелёным содержат отрицательные пары.

Симметричные		Асимметричные	
Датасет	**Размер**	**Датасет**	**Размер**
AllNLI [62],[56],[3]	282644	Miracl [32]	10000
MedNLI [43]	3699	MLDR [7]	1864
RCB [45]	392	Lenta [64]	185972
Terra [45]	1359	Mlsum [28]	51112
RuWANLI	35455	Mr-TyDi [33]	536600
Opus100 [20],[53]	1000000	Panorama [61]	11024
BiblePar [19]	62195	PravoIsrael [52]	26364
RudetoxifierDataDetox [30]	31407	Xlsum [57]	124486
RuParadetox [39]	11090	Fialka-v1 [1]	130000
Tapaco [49]	91240	RussianKeywords [31]	16461
RuHNP	500000	Gazeta [18]	121928
		Gsm8k-ru [60]	7470
		DSumRu [60]	27191
		SummDialogNews [60] [9]	75700
Всего положительных пар			3352653
Всего отрицательных пар			792644

ных пар).

В таблице 5 указаны все данные, используемые в дальнейшем обучении. Для датасетов: Gazeta, Mlsum, Xlsum, пары (заголовок/текст) и (заголовок/суммаризация) объединяются и используются в качестве асимметричных данных.

В итоге было собрано около 3.5 миллионов положительных пар и около 800 тысяч отрицательных. Каждый из этих наборов содержит 535 тысяч пар, сгенерированных мной. Все данные были приведены к унифицированному виду со столбцами: query, pos, neg. Для валидационной выборки я использовал валидационные части, предложенные авторами выбранных датасетов.

3. Эксперименты

На первом этапе используется предобученная модель *deepvk/deberta-v1-base*[5]. Далее следует этап **unlabeled** дообучения, на котором используется русская часть из корпуса *mMarco*. В рамках этого и последующих этапов отбираются негативные пары с помощью модели *baai/bge-m3*, которая генерирует одну отрицательную пару для каждого текста, у которого такая пара отсутствует.

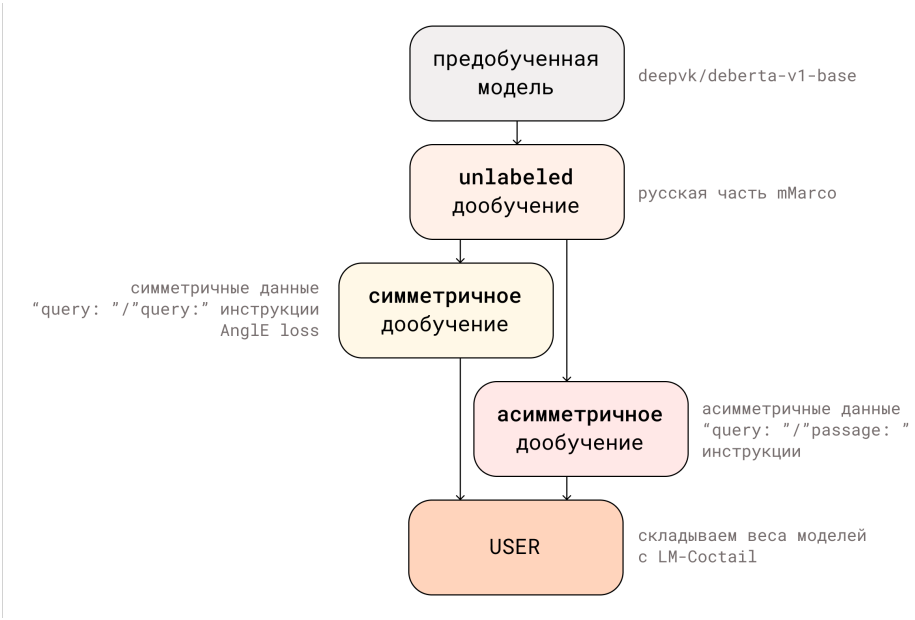


Рис. 8: Схема обучения модели USER

Второй этап был разделён на **симметричное**, усложнённое функцией потерь *AnglE*, и **асимметричное** дообучение. Каждому типу данных добавлялись соответствующие инструкции: *«query: »/«query: »* и *«query: »/«passage: »* соответственно. Затем обе модели обучались. По завершении обучения модели объединялись в итоговую с помощью *LM-Coctail* на основе данных из валидационной выборки.

Полученную модель я назвал **USER** (Universal Sentence Encoder for Russian). Схема обучения представлена на Рисунке 8.

[5]https://huggingface.co/deepvk/deberta-v1-base

3.1. Детали обучения

Поиск отрицательных пар выполнялся скриптом *hn_ mine* из репозитория[6]. Установленные значения параметров указаны в Таблице 6. Модель bge-m3 выбрана как наиболее точная. Отрицательные пары выбираются начиная от 10 ближайших, так как было замечено, что для симметричных датасетов, содержащих небольшие тексты, таких как Opus100, при меньших значениях этого аргумента, могут выбираться совпадающие тексты. Выюирается только одна отрицательная пара, так как большие значения искали пары, близкие к случайным.

Таблица 6: Значения параметров для поиска отрицательных пар.

Параметр	Значение
model_name_or_path	BAAI/bge-m3
range_for_sampling	10-210
negative_number	1

Для «unlabeled» и асимметричного дообучения я использовал скрипт *baai_ general_ embedding.finetune.run* из того же репозитория. Хоть для симметричной модели я и указывал 10 эпох обучения, но по результатам валидации был выбран 2500 шаг, то есть примерно 5-ая эпоха. Выбор максимальных длин токенов связан с тем, что 95 перцентиль соответсвующих данных находился внутри этого значения. Параметры обучения указаны в Таблице 7.

Асимметричная модель обучалась скриптом *train_ cli* из репозитория AnglE_emb[7]. В этом скрипте дополнительно указывались коэффициенты для функций потерь, при их выборе я руководствовался рекомендациями авторов статьи. Выбранные параметры указаны в Таблице 8. Заметим, что параметр **w1** равен нулю, то есть не используются случайный тексты, в качестве отрицательных пар, а только отрицательная пара из триплета.

Для объединения симметричной и асимметричной моделей в одну, я использовал LM-Coctail[8]. Веса подбирались на основе 5 примеров из валидаци-

[6]https://github.com/FlagOpen/FlagEmbedding/
[7]https://github.com/SeanLee97/AnglE/tree/main
[8]https://github.com/FlagOpen/FlagEmbedding/tree/master/LM$_C ocktail$

Таблица 7: Параметры для скрипта *baai_general_embedding.finetune.run*.

Параметр	unlabeled	асимметричное
learning_rate	1e-5	5e-6
fp16	True	True
num_train_epochs	1	10
per_device_train_batch_size	256	32
normlized	True	True
temperature	0.02	0.02
query_max_len	256	64
passage_max_len	256	512
train_group_size	2	2
sentence_pooling_method	mean	mean
gradient_accumulation_steps	1	32
query_instruction_for_retrieval	-	”query: ”
passage_instruction_for_retrieval	-	”passage: ”

Таблица 8: Параметры для скрипта *train_cli*.

Параметр1	Значение	Параметр2	Значение
w1	0	maxlen	128
w2	20	pooling_strategy	avg
w3	1	epochs	5
cosine_tau	20	batch_size	32
ibn_tau	20	gradient_accumulation_steps	4
angle_tau	20	fp16	True
learning_rate	1e-5	prompt_template	”query: {text}”

онной выборки и итоговые их значения равны 0.54 и 0.46 для симметричной и асимметричной моделей соответсвенно.

Большинтсво экспериментов проводилось на сервере с одной GPU NVIDIA Tesla A100 40 GB, однако «unlabeled» модель училась на кластере из 8 GPU NVIDIA A100 80GB.

4. Валидация моделей

Валидация полученной модели проводится на всех метриках, указанных в обзоре 1.3. Для метрик из «Энкодечки» рассчитвается **Mean S** – среднее значение по всем вышеуказанным задачам, и **Mean CLS** – среднее значение по задачам классификации для русского языка: TI, II, SA, IC.

Рассмотрим процесс валидации на каждой из выбранных метрик:

- «Энкодечка» содержит код валидации в официальном репозитории[9]. Для всех текстов я добавляю инструкцию «query: ».
- MIRACL использует библиотеку Pyseriny[10] для оценки моделей. Скрипт для MLDR находится в репозитории лаборатории baai[11]. Я добавляю асимметричные инструкции «query: » и «passage: » к запросам и документам, соответсвтенно.
- Задачи из MTEB доступны в одноимённой python библиотеке[12]. Она не предполагает указание инструкций, поэтому в этом случае я их не использую.

Каждый раз для оценки я указываю mean pooling и l2 нормализацию весов. Максимально число токенов равно максимальному их числу для моей модели – 512. Как и эксперименты, валидация проводилась с половинной точностью – fp16. При валидации на MLDR я устанавливаю максимальное число токенов равным 512 для всех моделей из-за ограничений по вычислительной мощности.

На задачах из MTEB мне не удалось провалидировать модель baai/bge-m3, так как она требует слишком много оперативной памяти (около 100 Гб на кластеризации). Поэтому в таблице она не указана.

[9]https://github.com/avidale/encodechka
[10]https://github.com/castorini/pyserini/blob/master/docs/prebuilt-indexes.md
[11]https://github.com/FlagOpen/FlagEmbedding
[12]https://github.com/embeddings-benchmark/mteb

4.1. Результаты

Сравним полученную модель на метриках 1.3.1, 1.3.2 с лучшими моделями из лидерборда «Энкодечка». Так как последний раз лидерборд обновлялся летом 2023 года, добавим к сравнению модели bge-m3 [7] и mult-e5-base [35]. Результаты представлены в таблицах 9, 10, 11.

Таблица 9: Результаты валидации модели на задачах «Энкодечки». Жирным выделена лучшая модель. Подчёркнута лучшая модель нашего размера. *Миллионов параметров без учёта эмбеддинг слоя.

Модель	Размер*	**Mean S**	**Mean CLS**	STS	PI	NLI	SA	TI	IA	IC	ICX
bge-m3	303	**0.786**	0.843	**0.86**	**0.75**	0.51	**0.82**	0.97	0.79	0.79	**0.78**
mult-e5-large	303	0.78	0.853	0.86	0.73	0.47	0.81	0.98	0.8	**0.82**	0.77
mpnet-base-v2	85	0.76	0.828	0.85	0.66	0.54	0.79	0.95	0.78	0.79	0.74
mult-e5-base	85	0.756	0.835	0.84	0.7	0.45	0.79	0.97	0.78	0.8	0.71
LaBSE-en-ru	85	0.74	0.815	0.79	0.66	0.43	0.76	0.94	0.77	0.79	0.77
sn-xlm-nli	85	0.74	0.77	0.76	0.6	**0.86**	0.76	0.9	0.72	0.7	0.6
Эта работа	85	0.772	**0.853**	0.85	0.74	0.48	0.81	**0.99**	**0.81**	0.8	0.7

Таблица 10: Результаты валидации модели на задачах извлечения релевантных текстов. Жирным выделена лучшая модель. Подчёркнута лучшая модель нашего размера. *Миллионов параметров без учёта эмбеддинг слоя.

			MIRACL		**MLDR**	
Модель	Размер*	Среднее	nDCG10	Recall100	nDCG10	Recall100
bge-m3	303	0.644	**0.678**	**0.959**	0.34	0.6
mult-e5-large	303	**0.651**	0.615	0.927	**0.402**	**0.66**
mpnet-base-v2	85	0.28	0.149	0.365	0.174	0.43
mult-e5-base	85	0.636	0.599	0.915	0.378	0.65
LaBSE-en-ru	85	0.28	0.1	0.327	0.209	0.485
sn-xml-nli	85	0.177	0.02	0.08	0.176	0.43
Эта работа	85	0.514	0.357	0.763	0.331	0.605

Полученная мной модель опережает все существующие модели того же размера по метрике **Mean S**, однако немного уступает большим моделям. Интересно, что на задачах классификации **Mean CLS** модель превосходит все другие модели, что свидетельствует о высоком качестве извлечения признаков текста. В задачах извлечения релевантных текстов модель показывает результаты хуже, чем уже существующие решения даже в своём размере. Однако на задачах из MTEB модель в среднем опережает остальные. Особенно примечательно, что в задачах кластеризации (P2P, S2S) модель зна-

Таблица 11: Результаты валидации на задачах из MTEB. Жирным выделена лучшая модель. Подчёркнута лучшая модель нашего размера. *Миллионов параметров без учёта эмбеддинг слоя.

		MTEB									
Модель	Размер*	Среднее	STS22	STSB	MSC	MIC	OPC	P2P	S2S	XNLI	XQR
mult-e5-large	303	0.695	0.631	0.819	**0.704**	0.65	0.89	0.416	0.399	0.773	**0.97**
mpnet-base-v2	85	0.675	0.587	**0.824**	0.699	0.632	0.904	0.363	0.37	**0.85**	0.845
mult-e5-base	85	0.675	0.615	0.791	0.671	0.62	0.87	0.414	0.403	0.741	0.95
LaBSE-en-ru	85	0.642	0.585	0.73	0.652	0.605	0.886	0.416	0.394	0.669	0.843
sn-xml-nli	85	0.639	0.512	0.729	0.576	0.519	0.866	0.416	0.399	1.0	0.735
Эта работа	85	**0.703**	**0.633**	0.815	0.69	**0.66**	**0.91**	**0.462**	**0.473**	0.766	0.92

чительно превосходит ближайшего конкурента, улучшив показатели с 0.416 до 0.462 и с 0.403 до 0.473 соответственно.

4.2. Анализ шагов обучения

Таблица 12: Сравнение двух стратегий «unlabeled» дообучения.

Модель	Энкодечка	MIRACL		MLDR
	Mean S	nDCG10	Recall100	(nDCG10)
user-unsup-mmarco-1-epoch	0.723	0.438	0.845	34.9
user-unsup-mmarco-wiki-plain	0.672	0.273	0.739	40.7

Оценив модели, полученные на этапе «unlabeled» дообучения, было решено использовать ту, что обучалась только на mMarco. Хоть на метрике MLDR она и проигрывает другой, зато на остальных опережает со значительным отрывом. Результаты представлены в таблице 12.

Оценим влияние шагов алгоритма 8 на метриках «Энкодечка» Mean S и MLDR $nDCG_{10}$. Результаты представлены в Таблице 13. Символом $\times$ обозначено отсутствие этапа в обучении. Расшифровка сокращённых названий этапов: ft – «unlabeled» дообучение; split – разделение на две модели; AnglE – добавление функции потерь AnglE; instr – использование инструкций.

Проанализировав данные в Таблице 13, можно отметить, что каждый последующий этап улучшений алгоритма приводит к росту обеих метрик. В частности, использование инструкций оказывает незначительное влияние на результаты, тогда как функция потерь AnglE, как и ожидалось, существен-

Таблица 13: Анализ шагов предложенного алгоритма.

Модель	«Энкодечка» Mean S	MLDR $nDCG_{10}$
USER × ft × split × AnglE × instr	0.743	0.267
USER × split × AnglE × instr	0.745	0.285
USER × AnglE × instr	0.76	0.326
USER × instr	0.769	0.328
USER	**0.772**	**0.331**

но улучшает качество на симметричных задачах. Наибольший прирост качества демонстрирует этап разделения по типу данных, что положительно сказывается на обеих метриках. Эти наблюдения подтверждают важность последовательного добавления улучшений в алгоритм для достижения оптимальных результатов.

Таким образом, собранные из открытых источников и сгенерированные нами данные, в сочетании с разработанным многоуровневым подходом обучения, позволили создать новую модель, сопоставимую с лидирующими решениями на рынке и даже превосходящую их по некоторым метрикам. В дальнейшем выработанные процессы можно будет переиспользовать для проведения новых экспериментов, например, для обучения модели большего размера. Также следует уделить внимание улучшению этапа «асимметричного» дообучения, чтобы повысить метрики выделения релевантных текстов.

Заключение

В ходе данной работы были достигнуты следующие результаты:

- Проведён обзор предметной области. Изучены подходы к обучению энкодеров текстов.
- Разработан алгоритм обучения энкодера текстов, использующий новейшие подходы. Предложено и реализовано несколько улучшений. Выполнена подготовка данных для обучения энкодера текстов. Собрано около 3 миллионов пар текстов из 23 датасетов. Дополнительно сгенерировано 2 датасета, содержащие ещё 540 тысяч положительных пар и столько же отрицательных.
- Обучена модель в соответствии с предложенным алгоритмом.
- Проведена апробация полученной модели. На целевой метрике удалось достичь роста качества среди моделей того же размера на 0.012 абсолютных единиц. Дополнительные метрики показали, что модель недостаточно эффективна в задачах извлечения релевантных текстов. Однако, в задачах кластеризации модель опережает все остальные с заметным отрывом, обгоняя ближайшую на 14%.

Список литературы

[1] 0x7o. fialka-v1. — 2024. — Access mode: `https://huggingface.co/datasets/0x7o/fialka-v1`.

[2] Zhong Ruiqi, Lee Kristy, Zhang Zheng, and Klein Dan. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections // arXiv preprint arXiv:2104.04670. — 2021.

[3] Nie Yixin, Williams Adina, Dinan Emily, Bansal Mohit, Weston Jason, and Kiela Douwe. Adversarial NLI: A new benchmark for natural language understanding // arXiv preprint arXiv:1910.14599. — 2019.

[4] Xiong Lee, Xiong Chenyan, Li Ye, Tang Kwok-Fung, Liu Jialin, Bennett Paul, Ahmed Junaid, and Overwijk Arnold. Approximate nearest neighbor negative contrastive learning for dense text retrieval // arXiv preprint arXiv:2007.00808. — 2020.

[5] Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N, Kaiser Łukasz, and Polosukhin Illia. Attention is all you need // Advances in neural information processing systems. — 2017. — Vol. 30.

[6] Devlin Jacob, Chang Ming-Wei, Lee Kenton, and Toutanova Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding // arXiv preprint arXiv:1810.04805. — 2018.

[7] Chen Jianlv, Xiao Shitao, Zhang Peitian, Luo Kun, Lian Defu, and Liu Zheng. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation // arXiv preprint arXiv:2402.03216. — 2024.

[8] Xiao Shitao, Liu Zheng, Zhang Peitian, and Muennighof Niklas. C-pack: Packaged resources to advance general chinese embedding // arXiv preprint arXiv:2309.07597. — 2023.

[9] CarlBrendt. Summ_Dialog_News. — 2022. — Access mode: https://huggingface.co/datasets/CarlBrendt/Summ_Dialog_News.

[10] Kashyap Abhinav Ramesh, Nguyen Thanh-Tung, Schlegel Viktor, Winkler Stefan, Ng See Kiong, and Poria Soujanya. A Comprehensive Survey of Sentence Representations: From the BERT Epoch to the CHATGPT Era and Beyond // Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers). — 2024. — P. 1738–1751.

[11] Dale David. Нейросети для Natural Language Inference (NLI): логические умозаключения на русском языке. — 2021. — October. — [Online; posted 10-October-2021]. Access mode: https://habr.com/ru/articles/582620/.

[12] Dale David. Перефразирование русских текстов: корпуса, модели, метрики. — 2021. — June. — [Online; posted 28-June-2021]. Access mode: https://habr.com/ru/post/564916/.

[13] Dale David. Рейтинг русскоязычных энкодеров предложений. — 2022. — June. — [Online; posted 5-June-2022]. Access mode: https://habr.com/ru/articles/669674/.

[14] Dale David. RuBERT Base Cased Paraphrase Detection by DeepPavlov. — 2024. — Access mode: https://huggingface.co/cointegrated/rubert-base-cased-dp-paraphrase-detection. Ported to Transformers format by cointegrated.

[15] Swayamdipta Swabha, Schwartz Roy, Lourie Nicholas, Wang Yizhong, Hajishirzi Hannaneh, Smith Noah A., and Choi Yejin. Dataset Cartography: Mapping and Diagnosing Datasets with Training Dynamics // Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) / ed. by Webber Bonnie, Cohn Trevor, He Yulan, and Liu Yang. — Online : Association for Computational Linguistics. — 2020. — Nov. — P. 9275–9293. — Access mode: https://aclanthology.org/2020.emnlp-main.746.

[16] Palangi Hamid, Deng Li, Shen Yelong, Gao Jianfeng, He Xiaodong, Chen Jianshu, Song Xinying, and Ward Rabab. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval // IEEE/ACM Transactions on Audio, Speech, and Language Processing. — 2016. — Vol. 24, no. 4. — P. 694–707.

[17] Karpukhin Vladimir, Oğuz Barlas, Min Sewon, Lewis Patrick, Wu Ledell, Edunov Sergey, Chen Danqi, and Yih Wen-tau. Dense passage retrieval for open-domain question answering // arXiv preprint arXiv:2004.04906. — 2020.

[18] Gusev Ilya. Dataset for Automatic Summarization of Russian News // Artificial Intelligence and Natural Language / ed. by Filchenkov Andrey, Kauttonen Janne, and Pivovarova Lidia. — Cham : Springer International Publishing. — 2020. — P. 122–134.

[19] Helsinki-NLP. BiblePara. — 2022. — Access mode: `https://huggingface.co/datasets/Helsinki-NLP/bible_para`.

[20] Zhang Biao, Williams Philip, Titov Ivan, and Sennrich Rico. Improving Massively Multilingual Neural Machine Translation and Zero-Shot Translation // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics / ed. by Jurafsky Dan, Chai Joyce, Schluter Natalie, and Tetreault Joel. — Online : Association for Computational Linguistics. — 2020. — July. — P. 1628–1639. — Access mode: `https://aclanthology.org/2020.acl-main.148`.

[21] Wang Liang, Yang Nan, Huang Xiaolong, Yang Linjun, Majumder Rangan, and Wei Furu. Improving text embeddings with large language models // arXiv preprint arXiv:2401.00368. — 2023.

[22] Inkoziev. Датасет перефразировок коротких фраз. — 2023. — Access mode: `https://huggingface.co/datasets/inkoziev/paraphrases`.

[23] Lau Jey Han and Baldwin Timothy. An empirical evaluation of doc2vec with practical insights into document embedding generation // arXiv preprint arXiv:1607.05368. — 2016.

[24] Le Quoc and Mikolov Tomas. Distributed representations of sentences and documents // International conference on machine learning / PMLR. — 2014. — P. 1188–1196.

[25] Laurer Moritz, Atteveldt Wouter van, Casas Andreu Salleras and Welbers Kasper. Less Annotating, More Classifying – Addressing the Data Scarcity Issue of Supervised Machine Learning with Deep Transfer Learning and BERT - NLI // Preprint. — 2022. — June. — Publisher: Open Science Framework. online; accessed: https://osf.io/74b8k (online; accessed: 2022-07-28).

[26] Li Xianming and Li Jing. Angle-optimized text embeddings // arXiv preprint arXiv:2309.12871. — 2023.

[27] Xiao Shitao, Liu Zheng, Zhang Peitian, and Xing Xingrun. Lm-cocktail: Resilient tuning of language models via model merging // arXiv preprint arXiv:2311.13534. — 2023.

[28] Scialom Thomas, Dray Paul-Alexis, Lamprier Sylvain, Piwowarski Benjamin, and Staiano Jacopo. MLSUM: The Multilingual Summarization Corpus // arXiv preprint arXiv:2004.14900. — 2020.

[29] May Philip. Machine translated multilingual STS benchmark dataset. — 2021. — Access mode: https://github.com/PhilipMay/stsb-multi-mt.

[30] Dementieva Daryna, Moskovskiy Daniil, Logacheva Varvara, Dale David, Kozlova Olga, Semenov Nikita, and Panchenko Alexander. Methods for detoxification of texts for the russian language // Multimodal Technologies and Interaction. — 2021. — Vol. 5, no. 9. — P. 54.

[31] Milana. russian_keywords. — 2024. — Access mode: https://huggingface.co/datasets/Milana/russian_keywords.

[32] Zhang Xinyu, Thakur Nandan, Ogundepo Odunayo, Kamalloo Ehsan, Alfonso-Hermelo David, Li Xiaoguang, Liu Qun, Rezagholizadeh Mehdi, and Lin Jimmy. Miracl: A multilingual retrieval dataset covering 18

diverse languages // Transactions of the Association for Computational Linguistics. — 2023. — Vol. 11. — P. 1114–1131.

[33] Zhang Xinyu, Ma Xueguang, Shi Peng, and Lin Jimmy. Mr. TyDi: A Multilingual Benchmark for Dense Retrieval // arXiv:2108.08787. — 2021.

[34] Müller Thomas, Pérez-Torró Guillermo, and Franco-Salvador Marc. Few-shot learning with siamese networks and label tuning // arXiv preprint arXiv:2203.14655. — 2022.

[35] Wang Liang, Yang Nan, Huang Xiaolong, Yang Linjun, Majumder Rangan, and Wei Furu. Multilingual e5 text embeddings: A technical report // arXiv preprint arXiv:2402.05672. — 2024.

[36] Su Hongjin, Shi Weijia, Kasai Jungo, Wang Yizhong, Hu Yushi, Ostendorf Mari, Yih Wen-tau, Smith Noah A, Zettlemoyer Luke, and Yu Tao. One embedder, any task: Instruction-finetuned text embeddings // arXiv preprint arXiv:2212.09741. — 2022.

[37] Maleki Farhad, Ovens Katie, Najafian Keyhan, Forghani Behzad, Reinhold Caroline, and Forghani Reza. Overview of machine learning part 1: fundamentals and classic approaches // Neuroimaging Clinics. — 2020. — Vol. 30, no. 4. — P. e17–e32.

[38] Pivovarova Lidia, Pronoza Ekaterina, Yagunova Elena, and Pronoza Anton. ParaPhraser: Russian paraphrase corpus and shared task // Conference on artificial intelligence and natural language / Springer. — 2017. — P. 211–225.

[39] Dementieva Daryna, Logacheva Varvara, Nikishina Irina, Fenogenova Alena, Dale David, Krotova Irina, Semenov Nikita, Shavrina Tatiana, and Panchenko Alexander. RUSSE-2022: Findings of the First Russian Detoxification Shared Task Based on Parallel Corpora.

[40] Reimers Nils and Gurevych Iryna. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. — Association for

Computational Linguistics. — 2019. — 11. — Access mode: http://arxiv.org/abs/1908.10084.

[41] Lewis Patrick, Perez Ethan, Piktus Aleksandra, Petroni Fabio, Karpukhin Vladimir, Goyal Naman, Küttler Heinrich, Lewis Mike, Yih Wen-tau, Rocktäschel Tim, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks // Advances in Neural Information Processing Systems. — 2020. — Vol. 33. — P. 9459–9474.

[42] Gao Yunfan, Xiong Yun, Gao Xinyu, Jia Kangxiang, Pan Jinliu, Bi Yuxi, Dai Yi, Sun Jiawei, and Wang Haofen. Retrieval-augmented generation for large language models: A survey // arXiv preprint arXiv:2312.10997. — 2023.

[43] Romanov Alexey and Shivade Chaitanya. Lessons from natural language inference in the clinical domain // arXiv preprint arXiv:1808.06752. — 2018.

[44] Martynov Nikita, Krotova Irina, Logacheva Varvara, Panchenko Alexander, Kozlova Olga, and Semenov Nikita. Rupaws: A russian adversarial dataset for paraphrase identification // Proceedings of the Thirteenth Language Resources and Evaluation Conference. — 2022. — P. 5683–5691.

[45] Shavrina Tatiana, Fenogenova Alena, Emelyanov Anton, Shevelev Denis, Artemova Ekaterina, Malykh Valentin, Mikhailov Vladislav, Tikhonova Maria, Chertok Andrey, and Evlampiev Andrey. RussianSuperGLUE: A Russian Language Understanding Evaluation Benchmark // arXiv preprint arXiv:2010.15925. — 2020.

[46] Bouscarrat Léo, Bonnefoy Antoine, Peel Thomas, and Pereira Cécile. STRASS: A light and effective method for extractive summarization based on sentence embeddings // arXiv preprint arXiv:1907.07323. — 2019.

[47] Saha Rohan. Influence of various text embeddings on clustering performance in NLP // arXiv preprint arXiv:2305.03144. — 2023.

[48] Sarkar Souvika, Feng Dongji, and Santu Shubhra Kanti Karmaker. Zero-Shot Multi-Label Topic Inference with Sentence Encoders // arXiv preprint arXiv:2304.07382. — 2023.

[49] Scherrer Yves. TaPaCo: A Corpus of Sentential Paraphrases for 73 Languages. — 2020. — Mar. — Access mode: https://doi.org/10.5281/zenodo.3707949.

[50] Vu Thanh, Nguyen Dat Quoc, Johnson Mark, Song Dawei, and Willis Alistair. Search personalization with embeddings // Advances in Information Retrieval: 39th European Conference on IR Research, ECIR 2017, Aberdeen, UK, April 8-13, 2017, Proceedings 39 / Springer. — 2017. — P. 598–604.

[51] Zhang Chi, Sah Shagan, Nguyen Thang, Peri Dheeraj, Loui Alexander, Salvaggio Carl, and Ptucha Raymond. Semantic sentence embeddings for paraphrasing and text summarization // 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP) / IEEE. — 2017. — P. 705–709.

[52] TarasHu. Q&A pairs relative to Israel's law in Russian language. — 2023. — Access mode: https://huggingface.co/datasets/TarasHu/pravoIsrael.

[53] Tiedemann Jörg. Parallel Data, Tools and Interfaces in OPUS // Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12) / ed. by Calzolari Nicoletta, Choukri Khalid, Declerck Thierry, Doğan Mehmet Uğur, Maegaard Bente, Mariani Joseph, Moreno Asuncion, Odijk Jan, and Piperidis Stelios. — Istanbul, Turkey : European Language Resources Association (ELRA). — 2012. — May. — P. 2214–2218. — Access mode: http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf.

[54] Cer Daniel, Yang Yinfei, Kong Sheng-yi, Hua Nan, Limtiaco Nicole, John Rhomni St, Constant Noah, Guajardo-Cespedes Mario, Yuan Steve,

Tar Chris, et al. Universal sentence encoder // arXiv preprint arXiv:1803.11175. — 2018.

[55] Liu Alisa, Swayamdipta Swabha, Smith Noah A, and Choi Yejin. Wanli: Worker and ai collaboration for natural language inference dataset creation // arXiv preprint arXiv:2201.05955. — 2022.

[56] Williams Adina, Nangia Nikita, and Bowman Samuel R. A broad-coverage challenge corpus for sentence understanding through inference // arXiv preprint arXiv:1704.05426. — 2017.

[57] Hasan Tahmid, Bhattacharjee Abhik, Islam Md. Saiful, Mubasshir Kazi, Li Yuan-Fang, Kang Yong-Bin, Rahman M. Sohel, and Shahriyar Rifat. XL-Sum: Large-Scale Multilingual Abstractive Summarization for 44 Languages // Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. — Online : Association for Computational Linguistics. — 2021. — Aug. — P. 4693–4703. — Access mode: https://aclanthology.org/2021.findings-acl.413.

[58] Conneau Alexis, Lample Guillaume, Rinott Ruty, Williams Adina, Bowman Samuel R, Schwenk Holger, and Stoyanov Veselin. XNLI: Evaluating cross-lingual sentence representations // arXiv preprint arXiv:1809.05053. — 2018.

[59] alenusch Anastasia Kozlova. Internal Fact-checking for the Russian language. — 2023. — Access mode: https://kaggle.com/competitions/internal-fact-checking-for-the-russian-language.

[60] d0rj. gsm8k-ru. — 2023. — Translated version of gsm8k dataset into Russian. Access mode: https://huggingface.co/datasets/d0rj/gsm8k-ru.

[61] its5Q. panorama. — 2022. — Dataset of satirical news from *Panorama*, Russian *The Onion*. Access mode: https://huggingface.co/datasets/its5Q/panorama.

[62] Bowman Samuel R, Angeli Gabor, Potts Christopher, and Manning Christopher D. A large annotated corpus for learning natural language inference // arXiv preprint arXiv:1508.05326. — 2015.

[63] Bonifacio Luiz Henrique, Jeronymo Vitor, Abonizio Hugo Queiroz, Campiotti Israel, Fadaee Marzieh, , Lotufo Roberto, and Nogueira Rodrigo. mMARCO: A Multilingual Version of MS MARCO Passage Ranking Dataset. — 2021. — 2108.13897.

[64] yutkin. Lenta.Ru-News-Dataset. — 2019. — Corpus of Russian news articles collected from Lenta.Ru. Access mode: `https://github.com/yutkin/Lenta.Ru-News-Dataset`.

ПРИЛОЖЕНИЕ А (страница 1 из 3)
Дополнительно о генерации RuWANLI

```
<instruction> Примеры:

<example 1 first sentence>
<label>: <example 1 second sentence>
<example 2 first sentence>
<label>: <example 2 second sentence>
<example 3 first sentence>
<label>: <example 3 second sentence>
<example 4 first sentence>
<label>: <example 4 second sentence>
<example 5 first sentence>
<label>: <example 5 second sentence>

Prompt for <category> generation:
<prompt>

Possible values for instructions:
```

Listing 1: Промпт для первоначальной генерации датасета RuWANLI. В таблице 14 указан промпт, который добавляется в зависимости от целевой категории.

Таблица 14: Зависящий от категории промпт.

Категория	**Инструкция**
contradiction	Написать 5 пар предложений, которые противоречат друг другу, как и предыдущие примеры.
entailment	Написать 5 пар предложений, как и предыдущие примеры. Второе предложение должно логически следовать из первого.
neutral	Написать 5 пар предложений, которые имеют такую же взаимосвязь, как и предыдущие примеры.

ПРИЛОЖЕНИЕ А (страница 2 из 3)

```
Я
 хочу, чтобы ты действовал в качестве генератора данных для NLI \
  ↪ датасета. Я буду передавать тебе текст, который я назову Q. \
  ↪ Для Q ты должен будешь сгенерировать E: логическое \
  ↪ следствие (entailment). E не должно просто быть \
  ↪ перефразированным текстом Q, используй более сложные связи. \
  ↪  Все сгенерированные тексты должны быть на русском языке. В \
  ↪ качестве ответа верни json с такой структурой: {"query": \
  ↪ текст< Q>, "entailment": текст< E>}.
\\ Примеры пар (Q, E):
1. {
 "query": "Я плачу сто двадцать один доллар в месяц у меня есть \
  ↪ еще один год, чтобы заплатить за мой дом.",
 "entailment": "Я плачу чуть больше 120 долларов в месяц."
};
2. {
 "query": "Прогресс Японии к парламентской демократии был \
  ↪ остановлен в x1930- годах растущим национализмом, \
  ↪ навязанным правительству генералами и адмиралами.",
 "entailment": "Рост национализма остановил продвижение Японии к \
  ↪ парламентской демократии."
}; 3. {
 "query": "Если мы решим остаться, многие люди умрут, но мы \
  ↪ надеемся, что сможем укусить бандитов на их пути.",
 "entailment": "Мы не можем остаться, потому что в противном \
  ↪ случае погибнет много мирных жителей."
}.
\\ Текст Q:
```

Listing 2: Промпт генерации entailment примеров для датасета RuWANLI.

ПРИЛОЖЕНИЕ А (страница 3 из 3)

```
Я
 хочу, чтобы ты действовал в качестве генератора данных для NLI \
  ↪ датасета. Я буду передавать тебе предложение, которое я \
  ↪ назову Q. Для Q ты должен будешь сгенерировать C: \
  ↪ противоречие (contradicton). C не должно просто быть \
  ↪ отрицанием Q, используй более сложные связи.  Все \
  ↪ сгенерированные тексты должны быть на русском языке. В \
  ↪ качестве ответа верни json с такой структурой: {"query": \
  ↪ текст< Q>, "contradiction": текст< C>}.
\\ Примеры пар (Q, C):
1. {
 "query": "Области, обслуживаемые дорогами, были застроены и, \
  ↪ как правило, переполнены в разгар лета.",
 "contradiction": "Застроенные районы наиболее переполнены в \
  ↪ мягкие зимние месяцы."
};
2. {
 "query": "За человеком, привязанным к веревкам, наблюдает \
  ↪ толпа.",
 "contradiction": "Человек не привязан к верёвкам."
};
3. {
 "query": "Мужчина стоит, читает газету и курит сигару.",
 "contradiction": "Мужчина сидит на скамейке."};Текст
\\ Q:
```

Listing 3: Промпт генерации contradiction примеров для датасета RuWANLI.

ПРИЛОЖЕНИЕ Б

Дополнительно о генерации RuHNP

```
Я
 хочу, чтобы ты действовал в качестве генератора данных. Я буду \
  ↪ передавать тебе текст query.  Для него сгенерируй 5 \
  ↪ перефразированных текстов pos: [p1, p2, ..., p5] и 5 \
  ↪ похожих текстов, но имеющих противоположный смысл neg: [n1, \
  ↪ n2, ..., n5]. Предложения должны быть на русском языке. В \
  ↪ качестве ответа верни следующую структуру: {"query": query, \
  ↪ "pos": pos, "neg": neg}.
\\ Пример: {
 "query": "Маленький мальчик шёл по лесу, было темно и страшно.",
 "pos": [
   "По лесу шёл маленький мальчик, было темно и страшно.",
   "Маленький мальчик шёл через лес, темно и страшно.",
   "По темному лесу шёл маленький мальчик, ему было страшно.",
   "Маленький парень шагал по лесу, темно и страшно.",
   "Маленький мальчик шёл по тёмному лесу, и ему было страшно."
 ],
 "neg": [
   "Большой мальчик шёл по лесу, было светло и спокойно.",
   "Маленькая девочка бежала через лес, было ярко и весело.",
   "Взрослый мужчина шагал по лесу, было темно и волнительно.",
   "Маленький мальчик бежал по лесу, было светло и радостно.",
   "По лесу шла стая весёлых птиц, и всё было светло и тихо."
 ]
}
\\ Текст Q для генерации:
```

Listing 4: Промпт генерации перефразированных текстов для датасета RuHNP.

ПРИЛОЖЕНИЕ В (страница 1 из 2)

Формулы оценки для задачи выделения релевантных текстов

nDCG (normalized Discounted Cumulative Gain) часто используется для оценки качества ранжирования в системах информационного поиска и рекомендаций. Она измеряет, насколько хорошо результаты ранжирования соответствуют релевантности документов для пользователя.

Метрика nDCG вычисляется следующим образом:

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{2^{\mathrm{rel}_i} - 1}{\log_2(i+1)} \tag{5}$$

$$\mathrm{IDCG}_p = \sum_{i=1}^{p} \frac{2^{\mathrm{rel}_i^*} - 1}{\log_2(i+1)} \tag{6}$$

$$\mathrm{nDCG}_p = \frac{\mathrm{DCG}_p}{\mathrm{IDCG}_p} \tag{7}$$

Сначала вычисляется DCG (Discounted Cumulative Gain) с использованием формулы 5, где rel_i — релевантность документа на позиции i в ранжированном списке, p — количество позиций, которое мы рассматриваем. Далее вычисляется IDCG (Ideal Discounted Cumulative Gain) — идеальное значение DCG для идеального ранжирования 6, где rel_i^* — релевантность документа на позиции i, но отсортированная в порядке убывания релевантности. Наконец, значение nDCG (normalized DCG) вычисляется по формуле 7. Значение nDCG находится в диапазоне от 0 до 1, где 1 — идеальное ранжирование.

ПРИЛОЖЕНИЕ В (страница 2 из 2)

Recall используется в задачах ранжирования для оценки того, насколько хорошо система поиска обнаруживает релевантные документы среди всех доступных документов. Для каждого запроса q в коллекции найдём p релевантных документов по версии модели. В формуле 8, r_q – истинно релевантный документ, R – p наиболее релевантных документов по версии модели, Q – все запросы в коллекции.

$$\text{Recall}_p = \frac{\sum_{q \in Q} \begin{cases} r_q \in R, 1; \\ r_q \notin R, 0. \end{cases}}{|Q|} \tag{8}$$

Метрика Recall принимает значение от 0 до 1, где 1 означает, что для каждого запроса был найден релевантный документ.