

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет информатики, математики и компьютерных наук

Кабанов Денис Сергеевич

Машинный перевод в режиме реального времени

Выпускная квалификационная работа - МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

по направлению подготовки

01.04.02 Прикладная математика и информатика

образовательная программа

«Интеллектуальный Анализ Данных»

Рецензент

доцент кафедры информационных систем

и технологий факультета ИМИКН НИУ

ВШЭ в Нижнем Новгороде

С.В. Павлов

Научный руководитель

аспирант НИУ ВШЭ

в Нижнем Новгороде

С.М. Досов

Нижний Новгород, 2025

Введение

В условиях развивающейся общей глобализации и интеграции иностранных специалистов с их ресурсами во все сферы жизни общества, будь то бизнес, образование или любое другое социальное взаимодействие — эффективная передача информации становится ключевым аспектом. Более того, общение через языковые барьеры теперь имеет решающее значение не только для понимания, но и для своевременного реагирования на постоянно меняющиеся условия современной жизни. Возможность мгновенно переводить устную и письменную речь облегчает беспрепятственное общение и способствует сотрудничеству и пониманию между различными культурами. А рост глобализированных отраслей, и распространение цифровых коммуникационных платформ за пределами своих стран, что требуют немедленного доступа к информации на нескольких языках, подчёркивает актуальность темы развития машинного перевода в реальном времени. Это подтверждается и активными исследованиями в данной тематике с середины прошлого десятилетия и по сей день [3][4]. К тому же такой перевод уже сейчас играет важную роль во многих ситуациях, будь то конференции, прямые трансляции и взаимодействие с клиентами, когда своевременные ответы имеют критическое значение [2]. Однако доступ к большинству переводчиков на данный момент требует интернет-подключения, что не всегда может быть доступно и удобно.

В данной работе решена задача машинного перевода с русского на английский и наоборот, с английского на русский, в real-time режиме. Решение этой задачи подразумевало под собой создание приложения, способного переводить текст в реальном времени, при этом не требующего от пользователя постоянного нахождения “в сети”. Данный факт делает

выпускную квалификационную работу важной не только в исследовательском плане, но ещё и в практическом.

Стоит также отметить, что весь инференс производился на CPU, так как полученными результатами планировалось охватить максимальное количество устройств, включая мобильные телефоны, у которых даже самые актуальные прототипы далеко не всегда имеют встроенные GPU. К тому же скорость работы нейронных сетей на устройствах с GPU на порядок выше по сравнению с аналогичным оборудованием, но без них, что свело бы возможности анализа ускорения моделей к минимуму.

В конечном итоге, машинный перевод в реальном времени — это не просто технологическая инновация; это фундаментальный инструмент для преодоления коммуникационных барьеров и содействия межкультурному диалогу в глобализированном обществе.

Структура выпускной квалификационной работы

Глава 1: Теоретическая часть.

Глоссарий

Постановка задачи

Обзор литературы

Глава 2: Практическая часть.

Анализ и обработка данных

Выбор архитектуры

Улучшение качества машинного перевода

Оптимизация модели

Сравнение качества работы в различных рантаймах

Демо-приложение

Выводы

Источники

Приложения

Глава 1: Теоретическая часть.

Глоссарий

- Датасет — набор данных. В задаче машинного перевода такой набор данных состоит как минимум из пар вида ‘текст на оригинальном языке’ и ‘перевод на интересующий язык’.
- Сэмпл — одна пара из датасета.
- Нейронная сеть — математическая модель, состоящая из искусственных нейронов, которые объединены в слои, взаимодействующие между собой для обработки данных.
- Линейный слой — один из основных слоёв нейронных сетей, представляющий собой матричное перемножение с добавлением смещения. Имеет вид $y = Ax + b$, где y — результат работы слоя, A — весовая матрица, b — bias (смещение), а x — матрица значений, пришедшие на вход слоя.
- Дообучение (fine-tuning) — процесс, при котором веса в слоях нейронной сети видоизменяются в соответствии с методом обратного распространения ошибки для улучшения качества предсказаний модели.
- Инференс (inference) — процесс запуска нейронной сети с целью получения ответа от неё на заранее переданные инструкции (например, для замера метрик), без какого-либо последующего дообучения.
- Бейзлайн (baseline) — модель, выбранная в качестве базовой для сравнительного анализа. Именно от неё будут отталкиваться дальнейшие попытки улучшения и с ней же сравниваться полученные результаты.

- Прунинг (pruning) — удаление (зануление) значений в матрицах весов модели для повышения эффективности нейросети, избавления от избыточных связей.
- Фреймворк — набор модулей и инструментов, объединённый в общую структуру (платформу) для обеспечения быстрой разработки ПО.
- Рантайм — среда выполнения.

Постановка задачи

Целью данной работы является создание приложения для машинного перевода пары языков английский-русский, не требующее подключение к интернету, с сравнительно хорошим качеством и временем отклика, близким к real-time на CPU.

Для достижения цели были поставлены следующие задачи:

- 1) Ознакомиться с передовой литературой и подходами в данной тематике;
- 2) Проанализировать существующие данные и обработать их в зависимости от рассматриваемых языков;
- 3) Провести исследование и выбор архитектуры;
- 4) Улучшить качество машинного перевода;
- 5) Оптимизировать модель с целью ускорения инференса;
- 6) Сравнить качество работы оптимизированных моделей в различных рантаймах;
- 7) Создать удобное приложение для взаимодействия пользователя с полученными моделями.

Обзор литературы

Существует несколько основных подходов к переводу, каждый из которых имеет свои достоинства и недостатки.

- Самый изученный и тривиальный — это ручной перевод людьми с одного языка на другой [5][6]. Данный вариант очень время затратный и дорогой для исполнения, однако он же имеет наилучшее качество. По очевидным причинам, такой подход не годится для использования в работе, связанной с переводом в реальном времени.

Следующие варианты уже являются более современным и интересными:

- Первый из них — это Rule-Based Machine Translation (Рис. 1). Этот подход основывается на лингвистических правилах и грамматике как исходного, так и целевого языка [7][8]. При нём система использует словари и набор правил для анализа и генерации текста, благодаря чему имеет высокую точность в специализированных областях, где правила могут быть четко определены. Однако, этот вариант требует значительных усилий для разработки правил и словарей, а также плохо справляется с идиомами и контекстом.

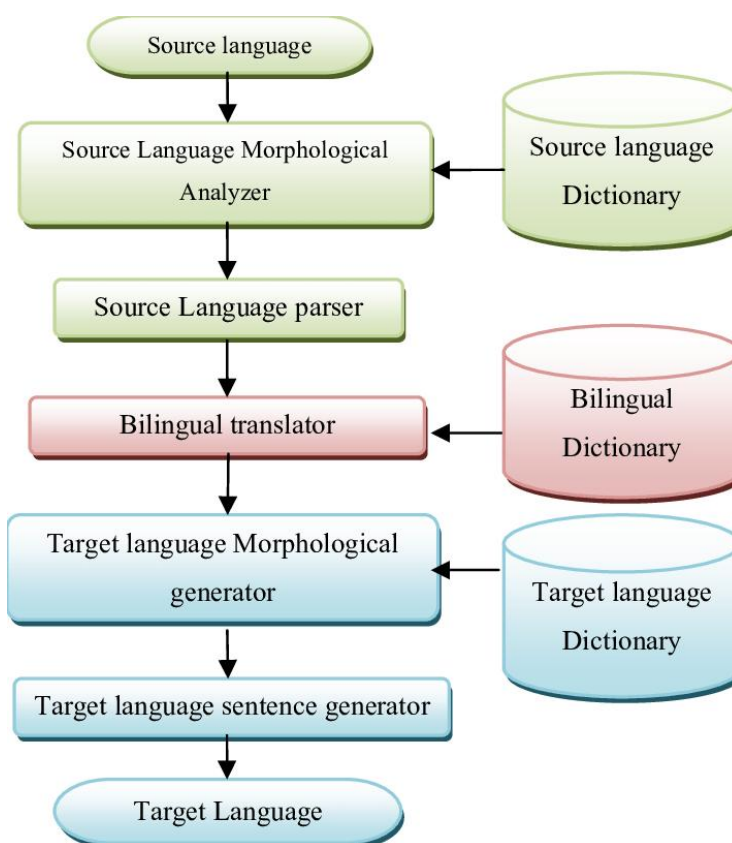


Рис. 1 - Концепция Rule-Based Machine Translation

- Следующий подход — это Statistical Machine Translation (Рис. 2). Он использует статистические модели для перевода текста [9][10], основанные на больших объемах двуязычных текстов, благодаря чему может автоматически адаптироваться к различным языкам и стилям.

Его разработка и поддержка уже не такие затратные, однако у него появляется новое бутылочное горлышко — необходимость большого объёма данных для покрытия различных последовательностей слов, так как иначе он может генерировать неестественные фразы.

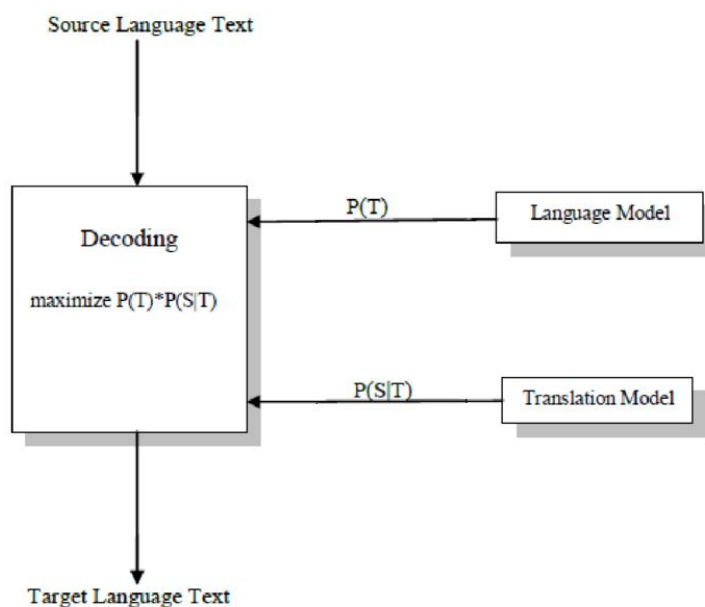


Рис. 2 - Концепция Statistical Machine Translation

- Самый же современный метод для перевода — это Neural Machine Translation. В этом подходе обучаются глубокие нейронные сети на больших корпусах данных, благодаря чему они могут учитывать контекст всего предложения, а не отдельных слов или фраз. Это способствует высокому качеству перевода, его естественности и плавности. Тем не менее даже у такого подхода есть свои недостатки, а именно — значительные требования к вычислительным ресурсам.

Среди описанных выше подходов самым быстрым и качественным на текущий день считается — перевод с использованием нейронных сетей, от RNN моделей (Recurrent Neural Network) [25], до Large Language Models (LLM), основанных на архитектуре Transformer [13][14][26]. Но так как суть работы заключается в получении относительно качественного перевода в режиме реального времени, то выбор падает не на огромные модели из

миллиардов параметров, а на небольшие модели-трансформеры, так как они легковесные и могут поместиться на многих устройствах, а также имеют быстрый инференс и сравнительно хорошее качество перевода. При этом, в отличие от RNN, трансформеры могут обрабатывать входящие последовательности параллельно, что ускоряет их инференс и обучение, лучше воспринимают долгосрочные зависимости в данных благодаря механизму внимания [13] и более устойчивы к шуму в них.

Глава 2: Практическая часть.

Анализ и обработка данных

В ходе исследования было рассмотрено 35 различных общедоступных датасета, представленных на сайте “Hugging Face”. Каждый из них включал как минимум одну пару языков, русский и английский, в некоторых таких пар было в разы больше, вплоть до полного покрытия 101-го языка. Среди проанализированных датасетов были такие известные в задаче машинного перевода представители, как WMT (вариации с 2014 по 2020 год) — один из основных бенчмарков для сравнения качества моделей машинного перевода many-to-many, OPUS, FLORES, ParaCrawl, UNPC, Tatoeba, TaPaCo. Каждый из них был выборочно просмотрен с целью изучения качества представленных языковых интерпретаций.

Как оказалось, почти все они имели свои недостатки, связанные с плохим качеством перевода с английского на русский (из-за подготовленности вида “множество-языков-в-множество-языков”), наличием “шумных” или мусорных сэмплов, слишком короткими переводами, состоящими только из пары слов или неудобным для работы представлением данных, где вместо перевода хранились раздробленные ссылки на онлайн хранилища.

Стоит отметить, что в работе рассматривается лишь перевод с русского на английский и обратно, с английского на русский, так как в открытом доступе практически отсутствуют датасеты с хорошей языковой интерпретацией, покрывающей сразу несколько обширных языков.

Тем не менее, в ходе продолжительного обзора был найден датасет “polynews-parallel”. Это многоязычный параллельный корпус данных, содержащий краткие сводки новостей с относительно качественным

переводом для 833 языковых пар. Он охватывает 64 языка, но для работы была взята лишь малая его часть, содержащая переводы с английского на русский, что всё же составила внушительные 176 441 сэмпла.



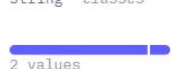
src string · lengths	tgt string · lengths	provenance string · classes
		
""Especially alarming is the high popularity of sweet soft drinks," the final report of the survey conclude...	""Особенную тревогу вызывает большая популярность сладких прохладительных напитков", говорится в заключительной части исследования."	wmtnews
As the main dish for lunch, pasta won very tightly over poultry.	В качестве основного обеденного блюда паста победила птицу с небольшим перевесом.	wmtnews
However, children showed very little interest in legumes, fish, and vegetable lunches.	Однако дети практически не проявили интереса к бобовым, рыбным и овощным блюдам.	wmtnews
The survey also showed that children are willing to adjust their eating habits provided they are given correct...	Исследование также показало, что дети готовы менять свои привычки в еде при условии, что им будет предоставлена соответствующая информация.	wmtnews
""Many participants from the survey control group showed a significant shift of preferences towards healthy ...	""Многие участники контрольной группы исследования продемонстрировали значительный сдвиг предпочтений в сторону здоровой пищи, когда заполня...	wmtnews
This was quite significant, for example, in case of sweet soft drinks, where more children would begin to pre...	Это было особенно заметно, например, в случае сладких прохладительных напитков - больше детей стали предпочитать им просто воду.	wmtnews

Рис. 3 - Пример данных в датасете “polynews-parallel”

После выбора подходящего для исследования датасета (Рис. 3), его предстояло обработать для дальнейшей передачи в обозреваемые решения. Данная обработка включала несколько простых шагов:

- 1) Удаление ненужного столбца ‘provenance’, отвечающего за оригинальный источник данных и не несущий полезной информации ни для исследования, ни для качества работы модели.
- 2) Попарное перемешивание значений в столбцах ‘src’ и ‘tgt’, так как в работе рассматривается задача многоязычного перевода (many-to-many), из английского в русский и из русского в английский. Таким образом при обучении и тесте у моделей на вход будет подаваться как текст на английском языке, так и на русском.
- 3) Добавление специфичного для каждой модели префикса к тексту в колонке ‘src’, отвечающего за то, на какой язык ожидается перевод. Например, для модели T5 данный префикс имеет вид “translate to ru: ”,

если исходный текст на английском и “translate to en: ”, если на русском.

- 4) Конвертация текста из колонок ‘src’ и ‘tgt’ в числовой формат (токены) с помощью BPE токенизатора [15], уникально подготовленного для каждой модели.
- 5) Разбиение на обучающую и тестовую выборки в пропорции 80 и 20 процентов соответственно. На обучающей части позже шёл fine-tuning весов модели, тогда как на тестовой — замер метрик.

Выбор архитектуры

Для рассмотрения возможных решений были выбраны модели на основе трансформеров, так как они показывают наилучшие метрики на многоязычных бенчмарках [16][17][18].

В качестве рассматриваемых решений-трансформеров было взято 6 моделей, пользовавшихся определённой популярностью, как на площадке Hugging Face (количество ежемесячных скачиваний), так и в среде исследователей [14][19][20][21] и имеющих сравнительно малый размер/количество параметров (что важно для скорости инференса, так как он должен быть real-time на CPU, и возможности уместить модели на не самые передовые устройства).

Эти модели: T5, mT5, MBart, M2M100, Marian и самая актуальная — LLaMA3.2, легковесная вариация LLaMA3, вышедшая в сентябре 2024 года. Все они были заранее предобучены на задачу генерации текста и поддерживают перевод как с английского на русский, так и с русского на английский. Сравнив их, будет выбрано одно решение для дальнейших исследований в качестве бейзлайна.

Как уже было сказано, все эти модели основываются на архитектуре трансформеров, принцип которой заключается в применении механизма “внимания” при обработке входящей последовательности [13]. Общими чертами всех моделей (Рис. 4), по-порядку вызова, является:

- 1) Применение эмбеддингов (Input Embedding) для кодирования приходящих токенов в векторное пространство некоторой размерности.
- 2) Сложение полученных эмбеддингов слов (токенов) с векторами, кодирующими позицию токена в изначальной последовательности (Positional Encoding).
- 3) N_e последовательных слоёв энкодера, включающих механизм “внимания” (Attention), нормализации (Norm) и полносвязные нейронные сети (Feed Forward), для создания контекстуализированных представлений токенов, которые учитывают их семантику и взаимосвязи в последовательности. Последовательное прохождение через слои позволяет модели постепенно уточнять и улучшать общий эмбеддинг входной последовательности (в моделях RNN под ним понималось скрытое состояние).
- 4) N_d последовательных слоёв декодера, схожих с теми, что были в энкодере (им теперь на вход поступает либо эмбеддинг стартового токена, например `<start>` или `<pad>`, либо последовательность сгенерированных токенов с предыдущей итерации декодера), но к ним также добавился механизм маскированного внимания (Masked Attention), что позволяет декодеру учитывать только предыдущие токены в выходной последовательности при генерации текущего токена и предотвращает использование информации о будущих токенах.
- 5) Выход последнего слоя декодера проходит через линейный слой (Linear) и функцию активации (Softmax), чтобы получить распределение вероятностей над словарем. Следующий токен выходной последовательности выбирается на основе этого распределения и процесс декодинга повторяется до тех пор, пока не будет сгенерирован токен конца последовательности, например, `<end>` или `</s>`.

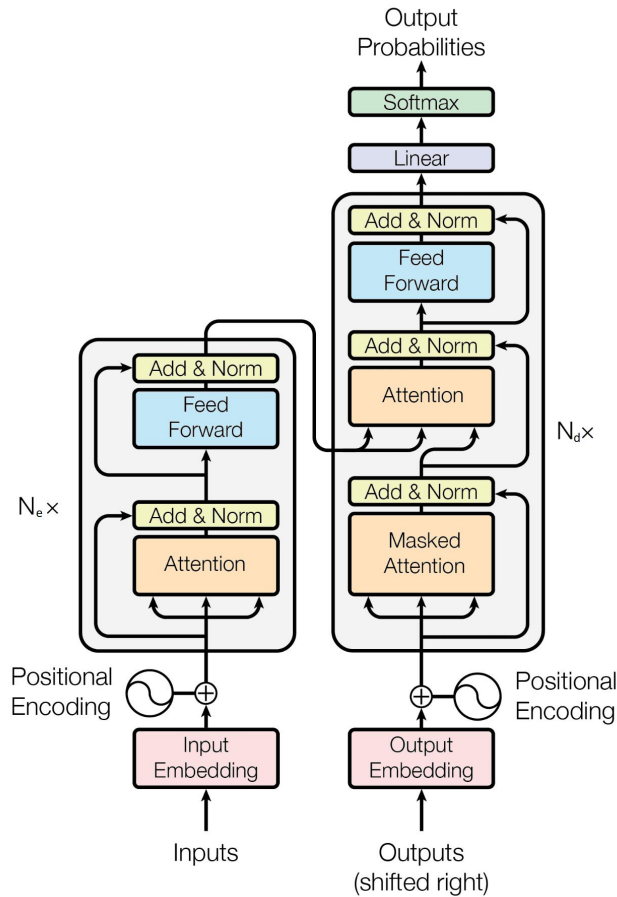


Рис. 4 - Стандартное строение моделей, основанных на архитектуре трансформеров

При этом каждое из решений старается внести в классическую архитектуру трансформеров определённые улучшения и оптимизации, например изменяя количество энкодеров и декодеров, размерности эмбеддингов, расположение нормализации до или после слоя внимания и полносвязной сети, тип используемого позиционного эмбеддинга и нормализации. Так, например, в T5 используется Pre-Layer Normalization вместо Post-LN, что помогает в решении проблемы затухания градиентов в глубоких сетях, вплоть до 30 слоёв [16], а в LLaMA3.2 классические синусоидальные позиционные эмбеддинги заменили на вращательное позиционное кодирование (Rotary Positional Embeddings, RoPE), которое более эффективно и лучше сохраняет информацию о расстоянии между токенами [22].

Сравнительный анализ представленных моделей был проведён с помощью подсчёта на тестовой части датасета двух основных метрик:

1) BLEU (Bilingual Evaluation Understudy) — метрика, оценивающая совпадения n -грамм в сгенерированной последовательности токенов и в одной или нескольких эталонных. BLEU принимает значения в диапазоне от 0 до 1, где чем выше значение, тем более качественный, похожий на эталонный, перевод генерирует модель. Ниже представлен принцип подсчёта:

- Для каждого n (обычно от 1 до 4) подсчитывается количество совпадающих n -грамм между сгенерированным переводом и эталонным переводом, обозначается это как:

$$p_n = \frac{\text{количество совпавших } n - \text{грамм}}{\text{общее количество } n - \text{грамм в сгенерированном переводе}}$$

- Вычисляется штраф (Brevity Penalty) за недостаток длины:

$BP = 1$, если длина сгенерированного перевода (c) больше длины эталонного (r)

$$BP = e^{1 - \frac{r}{c}}, \text{ если } c \leq r$$

- Общая BLEU-метрика вычисляется как взвешенное геометрическое точностей для разных значений n :

$$BLEU = BP * e^{\sum_{n=1}^N w_n * \log(p_n)}, \text{ где } N \text{ — максимальное максимальное значение } n \text{ (обычно 4), } w_n \text{ — веса для каждого } n \text{ (обычно равные, то есть } \frac{1}{N})$$

Современные многомиллиардные решения имеют значение BLEU в районе 0.4 ± 0.05 в зависимости от пары рассматриваемых языков, в данной же работе исследуются модели поскромнее, поэтому рекордных значений от них можно не ожидать.

2) Latency — задержка между отправкой запроса на перевод и получением ответа на него. Чем ближе значение к нулю, тем быстрее пользователь получает ответ от модели. Для оценки задержки при входных последовательностях различной длины использовалась линейная аппроксимация вида $ax + b$, где x — количество токенов входной последовательности, a — коэффициент наклона прямой, b — её смещение относительно начала координат.

Комбинация двух этих метрик позволит оценить, какая модель быстрее и точнее генерирует перевод.

Название модели	BLEU	Latency, sec	Количество обучаемых параметров (вес модели)
T5	0.25284	$0.06881 * x - 0.20910$	110 724 480
mT5	0.08152	$0.03518 * x + 0.00808$	244 309 248
MBart	0.25738	$0.23859 * x + 0.68243$	610 879 488
M2M100	0.23590	$0.18926 * x + 0.36501$	483 905 536
Marian	0.14473	$0.03513 * x + 0.16822$	66 546 176
LlaMA3.2 - 1B	0.00779	$0.13391 * x + 20.22426$	1 235 814 400

Табл. 1 - Сравнение метрик моделей машинного перевода (35000 сэмплов)

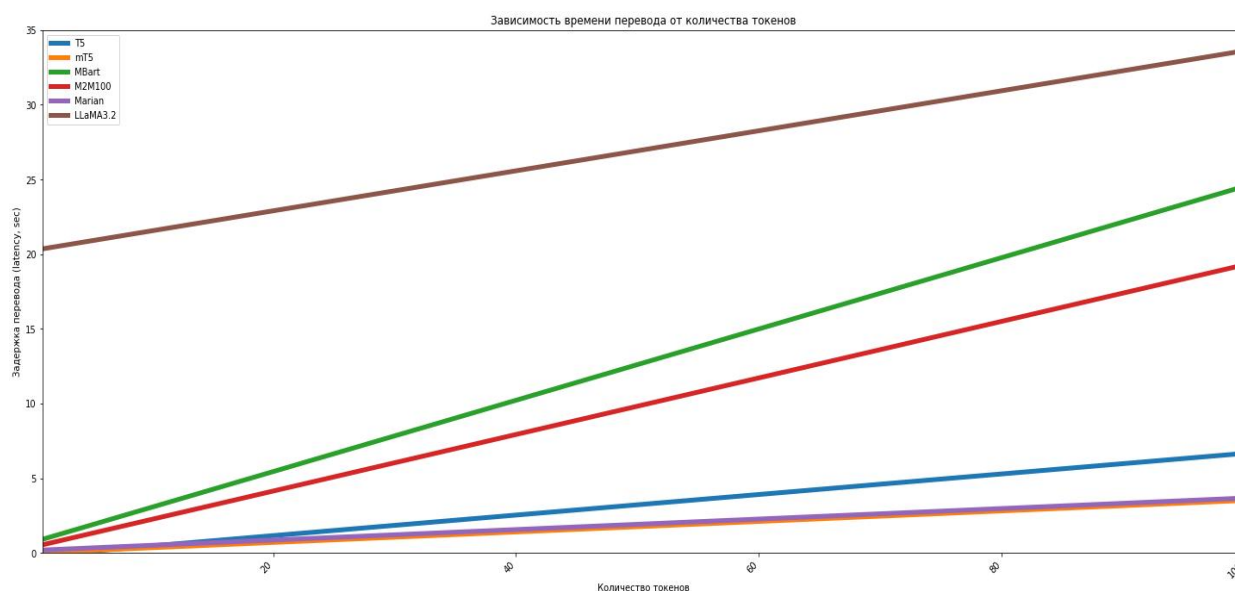


Рис. 5 - Сравнение Latency базовых моделей

Сравнивая получившиеся результаты замера метрик у моделей (Табл. 1 и Рис. 5), были получены следующие выводы:

- По метрике BLEU — MBart, T5, M2M100 оказались на голову выше своих конкурентов. У них примерно одинаковое значение меры, в районе 23 или 25 сотых. Худший результат оказался у LlaMA3.2, так

как её выход был сильно зашумлён ненужной информацией о процессе работы агента, но даже после его очистки, метрика осталась на низком уровне. Marian и mT5 также показали довольно слабые результаты.

- По Latency — модели Marian и mT5 продемонстрировали лучшую скорость работы. T5 оказалась чуть хуже них, тогда как M2M100, MBart, LLaMA3.2 показали самый медленный инференс.

Обобщив полученные показатели, предпочтение для дальнейшего анализа было отдано решению T5, так как оно помимо хороших показателей BLEU и Latency ещё и одно из самых легковесных, поэтому последующие эксперименты проходили именно с этой моделью.

Улучшение качества машинного перевода

Следующим этапом анализа была попытка улучшения качества перевода на подготовленном ранее датасете “polynews-parallel”.

Далее и все последующие дообучения шли на 80% данных, что составили примерно 141 000 пар переводов. Обучение шло максимум 20 эпох с возможным досрочным прерыванием при условии, что отслеживаемая метрика BLEU на тестовой части датасета (оставшиеся 20%) не улучшилась в течении трёх эпох. Прерывание при тесте нужно, чтобы избежать больших бессмысленных временных затрат, когда модель уже “изучила” все знания из обучающих данных и начинает переобучаться, то есть досконально заучивать ответы лишь для улучшения метрик на тренировочной части. Learning rate при этом динамически изменялся, начиная с $1e-5$ (0.00001) с последующим уменьшением, что способствовало улучшению сходимости модели [23].

Результаты такого обучения решения T5 представлены ниже (Рис. 6 и 7). По ним видно, что обучение шло 19 эпох из 20 (сработал критерий досрочной остановки), при этом лучшая модель по метрике BLEU на тестовой части была получена на эпохе №16 (средний график рисунка), после чего BLEU вышел на своеобразное плато. Также можно подметить, что целевая функция (Cross-Entropy Loss, левый график на рисунке) продолжала уменьшаться на тренировочной части набора данных даже к концу обучения, тогда как на тестовой части она достигла своего оптимума

примерно к 10-11 эпохе, после чего лишь колеблется возле значения 0.327. Среднее количество генерируемых токенов (правый график) изменялось слабо, и за все эпохи почти не сдвинулось со значения в 29 токенов. Время такого дообучения составило 63.8 часа.

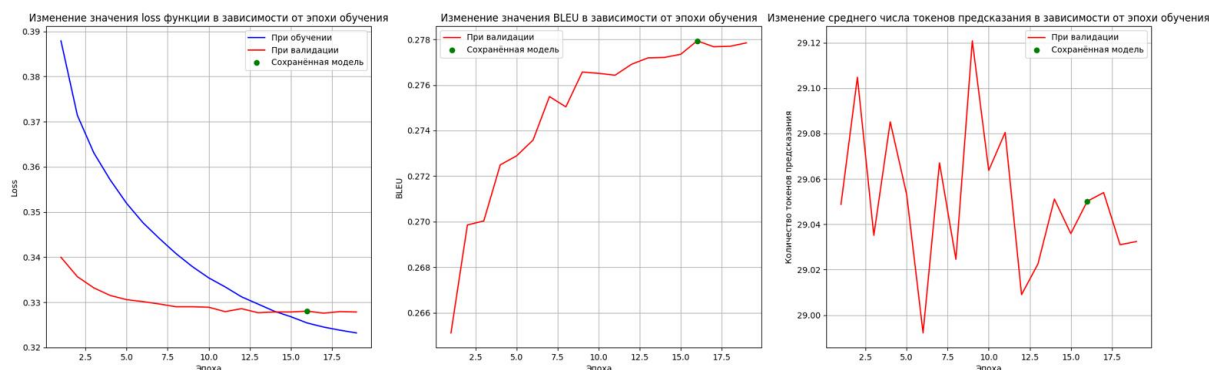


Рис. 6 - Изменение отслеживаемых метрик в процессе обучения, слева направо: Cross-Entropy Loss, BLEU, Среднее количество генерируемых токенов

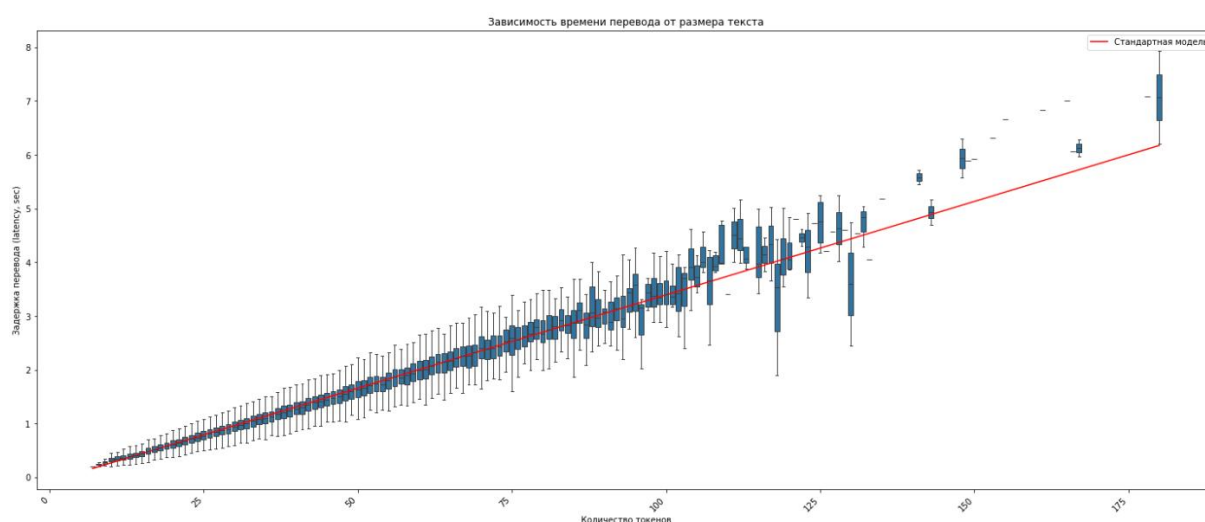


Рис. 7 - Зависимость Latency от размера входа для дообученной модели T5

Подытожив, дообучение привело к увеличению метрики BLEU на ~10% (с изначального 0.25284 до 0.27794) при рассмотрении эпохи № 16. Поэтому, чтобы добиться максимального улучшения по сравнению с изначальной моделью, веса модели именно с этой эпохи будут использовать на следующем шаге, на этапе сжатия.

Оптимизация модели

На этапе оптимизации модели основной задачей было — ускорить её инференс, при этом не сильно потеряв в метрике BLEU. Для достижения этой цели без серьёзных вмешательств в архитектуру решения существует несколько техник — квантизация, факторизация матриц и прунинг.

- Квантизация (квантование) — метод сжатия весов нейронной сети, который позволяет хранить их в более компактном виде посредством снижения точности представления (например, с 32-битного float до 8-битного int числа).
- Факторизация матриц — метод, используемый для разложения матриц весов на произведение двух или более матриц меньшего размера.
- Прунинг — удаление незначительных весов или нейронов из сети, что позволяет уменьшить размер модели и ускорить инференс без значительной потери точности.

Для исследования был выбран последний вариант — прунинг, так как среди представленных техник он имеет самый гибкий функционал, что позволяет применять его к любой архитектуре нейронной сети, а также наименьшее влияние на качество перевода, потому что он удаляет (зануляет) незначительные веса. К тому же его применение не влечёт за собой сложных изменений в алгоритмах для повторного обучения и инференса.

Если рассматривать доступные в фреймворке PyTorch варианты прунинга, то их можно разделить на два основных типа — неструктурированный и структурированный прунинг. Оба они лишь зануляют заданный процент весов **линейных слоёв** модели, что имеют наименьшие значения L1 или L2 нормы, без полного их удаления (то есть без изменения размерностей). Отличаются эти подходы лишь тем, каким образом выбираются параметры для прунинга. В неструктурированном варианте зануляются отдельные элементы весовых матриц, тогда как в структурированном они выбираются “блоками”, то есть значения в матрицах весов при таком подходе зануляются целыми строками или столбцами.

Среди рассмотренных вариаций прунинга были следующие комбинации:

- Не структурированный прунинг: 20, 33 и 50% параметров.
- Структурированный прунинг: 10, 15, 20 и 25% параметров. Данный вариант сильно сказывался на генеративных способностях модели, что зачастую приводило к серьёзному ухудшению качества предсказаний, когда первые токены последовательности переводились правильно, после чего модель уходила в циклическое повторение уже выданных комбинаций.

Важно отметить, что при подсчёте Latency учитывалось не только время генерации ответа модели, но ещё и время на токенизацию входной последовательности. Однако, в данном случае им можно пренебречь, так как оно составляет очень малую часть от общего времени перевода T5 модели (десятитысячные доли секунды). Но если бы рассматривались другие решения, с большим количеством возможных токенов (в T5 их всего 65 000), то тут потребовался бы дополнительный анализ.

Эксперименты с сжатием модели без дальнейшего дообучения привели лишь к ухудшению метрик, как BLEU, так и Latency. Падение Latency в данном случае связано с тем, что предсказание токена конца строки `<eos>` было нарушено, из-за чего генерировались более нестабильные по количеству токенов и не имеющие смысла переводы. Как итог, ухудшение обеих метрик.

Поэтому было решено провести дополнительное дообучение с целью улучшения метрик сжатых моделей. При этом занулённые при прунинге веса остались неизменёнными, так как к ним применилась маска, из-за чего градиент для них равнялся нулю, а следовательно и обновление таких весов на повторной тренировке не производилось. Результаты сжатия и дообучения представлены ниже (Табл. 2, Рис. 8). В колонке “Вариант модели” под ‘Стандартная модель’ понимается изначально взятое решение T5 с Hugging Face, ‘без прунинга’ — её дообученная версия, все последующие варианты — сжатые и снова натренированные веса модели ‘без прунинга’.

Вариант модели	BLEU (сравнительная позиция)	Latency, sec (сравнительная позиция)
стандартная модель	0.25284 (9)	$0.03474 * x - 0.08122$ (7)
без прунинга	0.27794 (2)	$0.03474 * x - 0.08122$ (6)
неструктурированный, 20%	0.27813 (1)	$0.02900 * x - 0.02297$ (3)
неструктурированный, 33%	0.27425 (3)	$0.02862 * x - 0.01597$ (2)
неструктурированный, 50%	0.26542 (6)	$0.03190 * x - 0.02153$ (4)
структурированный, 10%	0.27252 (4)	$0.04025 * x - 0.05476$ (9)
структурированный, 15%	0.26837 (5)	$0.03870 * x - 0.02747$ (8)
структурированный, 20%	0.26402 (7)	$0.03319 * x + 0.02051$ (5)
структурированный, 25%	0.25835 (8)	$0.02471 * x - 0.00604$ (1)

Табл. 2 - Сравнение метрик после прунинга (35000 сэмплов)

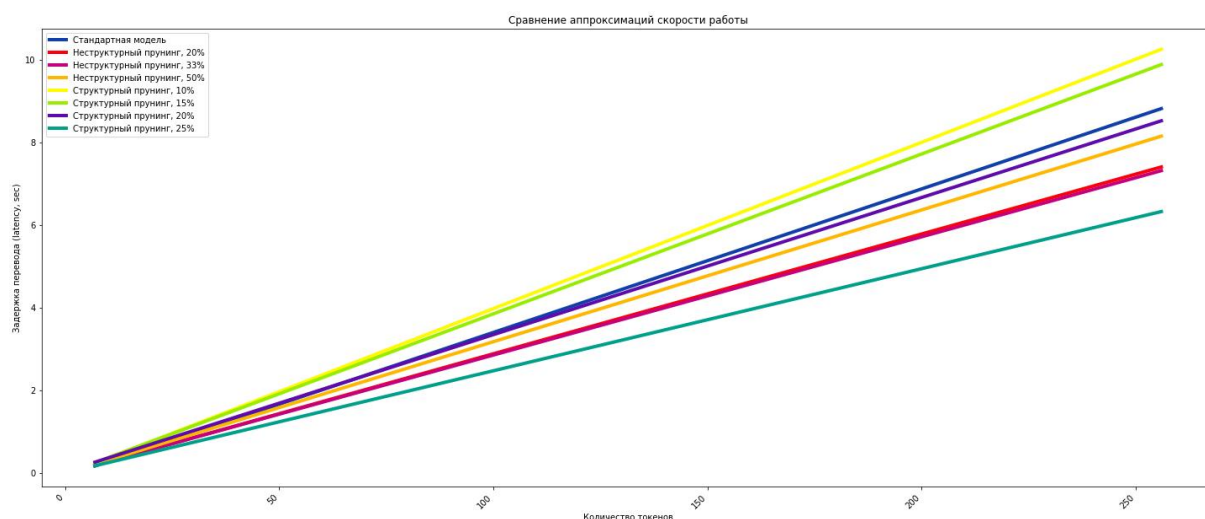


Рис. 8 - Сравнение аппроксимаций Latency вариантов прунинга

Анализируя полученные результаты, было замечено, что при одном из вариантов прунинга (неструктурированный, 20% параметров) значение BLEU даже слегка увеличилось, на 0.00019 по сравнению с моделью без занулённых весов (0.27794 → 0.27813). Это явление можно назвать

случайной погрешность, так как улучшение составляет менее одной десятой процента, тем не менее это означает, что для каких-то примеров в тестовой части зануление весов модели пошло на пользу генерализации знаний [24]. Для остальных вариантов прунинга — метрика BLEU ухудшилась, разберём их подробнее.

- Для неструктурированного прунинга 33% весов (BLEU=0.27425), структурированного 10% (BLEU=0.27252) и 15% (BLEU=0.26837) метрика упала не сильно и имеет значение ближе к полной натренированной модели (BLEU=0.27794), чем к её изначальной версии, взятой с Hugging Face (BLEU=0.25284).
- Для неструктурированного прунинга 50 процентов параметров линейных слоёв (BLEU=0.26542), структурированного 20% (BLEU=0.26402) и 25% (BLEU=0.25835) метрика уже ближе к изначальным значениям (BLEU=0.25284), но она всё ещё выше, из чего следует вывод, что если занулить большой, в разумных пределах, процент весов модели и снова дообучить её — она всё ещё будет лучше необученной вариации.

Если рассматривать метрику Latency, то тут уже не всё так однозначно. Большинство вариантов получили ускорение инференса от зануления весов, но не все. Так при структурированном прунинге 10 и 15 процентов весов линейных слоёв произошло замедление инференса моделей примерно на 13%, тогда как для всех остальных вариантов наблюдалось лишь ускорение.

- Самый быстрый инференс показало решение с структурированным прунингом 25-ти процентов весов, где ускорение при подсчёте метрик оказалось аж на 27%, что может быть связано с генерацией последовательностей самой малой длины, в среднем - 28.64 токена, однако при этом метрика BLEU данного решения — худшая в дотренированных вариациях.
- Следующими по ускорению инференса идут варианты с неструктурированным прунингом 20-ти и 33-ти процентов, у них ускорение оказалось примерно на 17.5%, при этом BLEU практически не ухудшился. Данные два решения показали наилучшие соотношения потери BLEU к ускорению, но для дальнейшего анализа будет выбрано лишь одно из них — модель с занулёнными 20-ю процентами весовых матриц.

- Структурный прунинг 20-ти процентов и неструктурный 50-ти показали средние результаты по ускорению, 3% и 8% соответственно. BLEU же у них ближе к изначальной модели, чем к дообученной.

Таким образом зануление значений в весовых матрицах может как ускорить инференс ценой потери качества, так и замедлить при неудачных комбинациях обнулённых параметров.

Ниже представлены подробные графики дообучения и зависимости Latency от длины входной последовательности выбранного для дальнейшего анализа варианта модели (Рис. 9 и 10). Графики при остальных вариациях прунинга представлены в приложениях (Рис. 13-24).

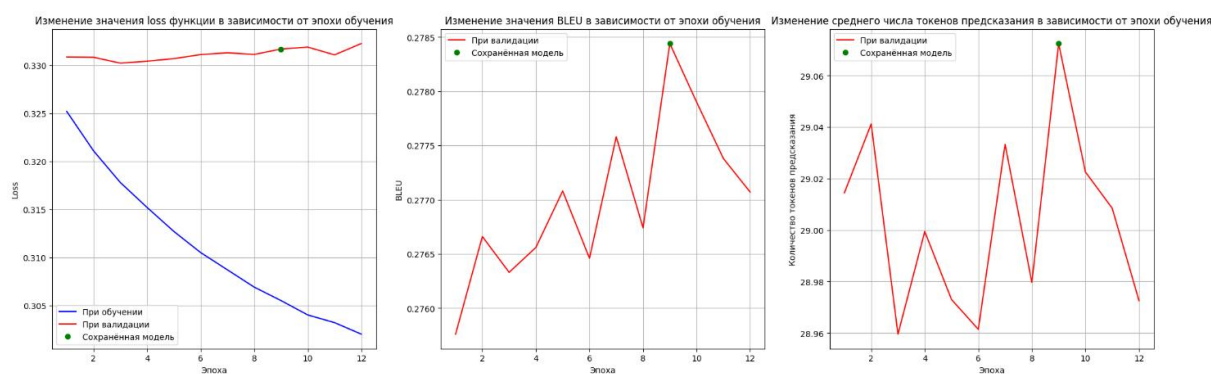


Рис. 9 - Изменение отслеживаемых метрик в процессе обучения для модели T5 с неструктурированным прунингом 20% параметров, слева направо: Cross-Entropy Loss, BLEU, Среднее количество генерируемых токенов

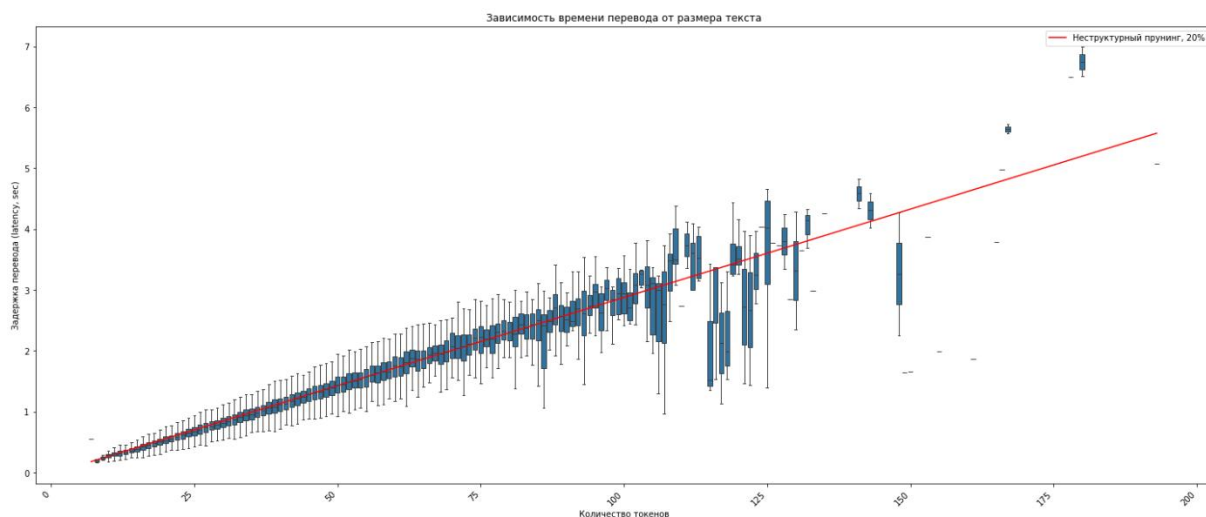


Рис. 10 - Зависимость Latency от размера входа для дообученной модели T5 с неструктурированным прунингом 20% параметров

Сравнение качества работы в различных рантаймах

Предпоследним шагом было сравнение качества работы модели при её запуске в различных рантаймах, чтобы выявить наиболее эффективные платформы для исследуемой модели, которые позволят ещё сильнее ускорить её инференс. Среди рассмотренных были следующие рантаймы:

- ONNX: самый универсальный рантайм, предназначенный для ускорения и упрощения развертывания моделей из различных фреймворков, а также на разных платформах и устройствах (Windows, Linux, Mac, GPU, CPU, мобильные устройства).
- openVINO: рантайм, специально разработанный для ускорения моделей на процессорах Intel, а также FPGA.
- PyTorch: базовый рантайм (фреймворк), что широко используется в научных исследованиях и промышленности благодаря своему простому интерфейсу взаимодействия и наличию инструментария для проведения различных экспериментов, от обычного обучения, до и изменения структуры решений.
- ExecuTorch: рантайм, специально ориентированный на быстрый и оптимизированный запуск уже обученных моделей из PyTorch в продакшене на мобильных и периферийных устройствах, от высокопроизводительных телефонов до жестко ограниченных встроенных систем и микроконтроллеров.

Для анализа на этом шаге была выбрана лучшая нейронная сеть, полученная после прунинга весов, то есть T5 с прунингом 20% весов. Сравнение как и раньше проводилось на CPU (более медленном, представленном в Google Colab), но теперь лишь на 5000 тестовых сэмплов (до этого их было 35 000), так как реализованные эксперименты показали, что метрика Latency (её линейная аппроксимация) практически не изменяется при увеличении количества рассматриваемых пар выше 5000.

Анализ качества работы для всех рантаймов имел приблизительно одну и ту же структуру, состоящую сначала из конвертации лучшей модели с предыдущего шага (написанной в фреймворке PyTorch) в выбранный рантайм, после чего шло приведение её в режим инференса и замер

скорости работы. Результаты рассмотрения рантаймов представлены ниже (Табл. 3).

Название рантайма	BLEU	Latency, sec
ONNX	0.27558	$0.01853 * x - 0.01955$
openVINO	0.27566	$0.02187 * x - 0.00335$
PyTorch	0.27558	$0.04854 * x - 0.01646$
ExecuTorch	0.27302	$0.05841 * x - 0.25661$

Табл. 3 - Сравнение метрик в различных рантаймах (5000 сэмплов)

При конвертации из одного рантайма в другой, BLEU, как и ожидалось, немного разнится (но всё равно остаётся на уровне дообученной модели). Это обусловлено тем, что у каждого рантайма своя реализация операций, построение графов вычислений и методы взаимодействия с весами.

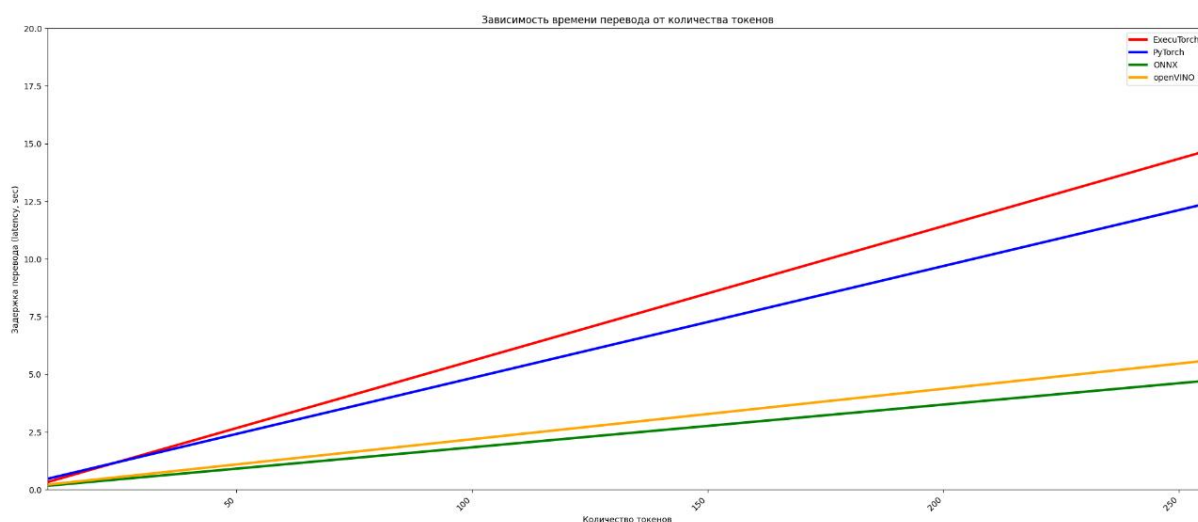


Рис. 11 - Сравнение аппроксимаций Latency вариантов рантайма

Сравнивая задержку получения ответа от модели (Рис. 11), было обнаружено, что вычисления в ExecuTorch, замедлились по сравнению с изначальным рантаймом PyTorch приблизительно на 20%, тогда как в рантаймах openVINO и ONNX они значительно ускорились, более чем в два раза в openVINO и почти в три раза в ONNX, на 55% и 62% соответственно. Однако, цена этого ускорения — увеличение потребляемой памяти, так

модель в рантайме openVINO стала весить на 30% больше (423 Mb → 551 Mb), а в ONNX аж на 110% (423 Mb → 889 Mb), что более чем в два раза.

Данные наблюдения можно охарактеризовать трейд-оффом памяти на скорость исполнения. Таким образом, если пользователю нужна будет самая быстрая вариация модели и он не имеет строгих ограничений по памяти, то openVINO или ONNX будет его выбором, иначе лучше остаться на PyTorch.

Демо-приложение

Для создания приложения был выбран фреймворк Streamlit, позволяющий развёртывать модели машинного обучения в красивых веб-приложениях (Рис. 12). Основной функционал приложения включает в себя как выбор модели из описанных в работе, так и языка, на который требуется сделать перевод (английский или русский). Когда решение и языки будут выбраны, пользователь может ввести текст в появившееся поле и запросить от модели языковую интерпретацию в реальном времени. Через несколько мгновений ему придёт перевод вместе со сгенерированным графиком, динамически отслеживающим эффективность, то есть скорость работы модели.

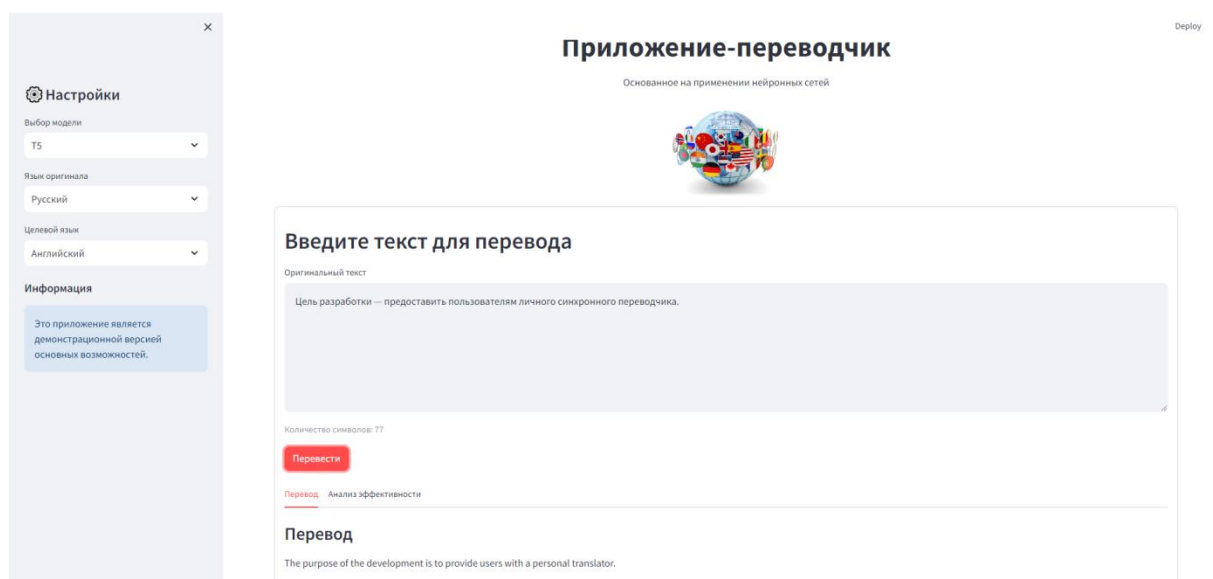


Рис. 12 - Интерфейс взаимодействия с приложением

Выводы

В рамках данной дипломной работы была исследована актуальная проблема машинного перевода в реальном времени, что является важным направлением развития современных технологий обработки естественного языка. Согласно изначально поставленным задачам, было рассмотрено несколько современных решений задачи нейронного машинного перевода (NMT), имеющих в своей основе архитектуру трансформеров (T5, mT5, MBart, M2M100, Marian, LLaMA3.2). Проведено их сравнение по двум метрикам — BLEU и Latency, после которого была определена лучшая модель, как бейзлайн для дальнейшей работы. С этой моделью были проведены различные эксперименты, включающие:

1. Дообучение, благодаря которому удалось увеличить метрику качества перевода на ~10%.
2. Прунинг (зануление весов с последующим обучением для восстановления потерянных моделью свойств), ускоривший инференс модели почти на 27%, но значительно ухудшивший качество перевода, из-за чего была выбрана модель с более скромным ускорением (17.5%), но почти не потерявшая в BLEU.
3. Сравнение работы в различных рантаймах, что позволило ускорить модель ещё почти в три раза (на 62%) ценой увеличения занимаемой памяти.

Финальным этапом была разработка приложения на Streamlit, позволяющего пользователю легко взаимодействовать с рассмотренными моделями.

Созданные в ходе работы прототипы и модели демонстрируют потенциал для применения в реальных условиях, таких как международное общение или мультязычные сервисы, что способствуют ускорению межкультурного взаимодействия и делает технологии более доступными. В целом, развитие систем машинного перевода в реальном времени сейчас находится на стадии активного исследования и поэтому обладает значительным потенциалом для дальнейших улучшений. Применительно к данной ВКР, научный интерес имеют следующие варианты её развития: рассмотрение оставшихся техник оптимизации модели (квантизация и факторизация матриц), увеличение количества поддерживаемых языков или попытка перехода в мультимодальность (работа не только с текстом).

Источники

- [1] **F. Calefato, F. Lanubile, T. Conte, R. Prikladnicki.** Assessing the impact of real-time machine translation on multilingual meetings in global software projects. *Empirical Software Engineering* 21(3), 2015.
- [2] **F. Calefato, F. Lanubile, P. Minervini.** Can Real-Time Machine Translation Overcome Language Barriers in Distributed Requirements Engineering?. *5th IEEE International Conference on Global Software Engineering, ICGSE*, 2010.
- [3] **Jiatao Gu, G. Neubig, Kyunghyun Cho, Victor O.K. Li.** Learning to Translate in Real-time with Neural Machine Translation. *15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1053-1062, 2017.
- [4] **E. Nurvitadhi, M. Naik, A. Boutros, P. Budhkar, A. Jafari, Dongup Kwon, D. Sheffield, A. Prabhakaran, K. Gururaj, P. Appana.** Scalable Low-Latency Persistent Neural Machine Translation on CPU Server with Multiple FPGAs. *International Conference on Field-Programmable Technology (ICFPT)*, 2019.
- [5] **L. Venuti.** The Translator's Invisibility: A History of Translation. *The Journal of English and Germanic Philology* Vol. 96, No. 1, pp. 71-73, 1997.
- [6] **J. Hutchins.** Machine Translation and Human Translation: In Competition or in Complementation? *International Journal of Translation*, vol.13, no.1-2, 2001.
- [7] **Hiroyuki Kaji.** An Efficient Execution Method for Rule-Based Machine Translation. *12th conference on Computational linguistics - Volume 2*, pages 824 - 829, 1988.
- [8] **M. Carl, C. Pease, O. Streiter.** Linking Example-Based and Rule-Based Machine Translation. *Article from Institute for Applied Information Research, Germany*, 1999.
- [9] **A. Lopez.** Statistical machine translation. *ACM Computing Surveys (CSUR)*, Volume 40, Issue 3, Article No.: 8, Pages 1 - 49, 2008.
- [10] **M. Hearne, A. Way.** Statistical Machine Translation: A Guide for Linguists and Translators. *Language and Linguistics Compass* 5(5):205-226, 2011.

- [11] **L. Medsker, L. Jain.** Recurrent Neural Networks: Design and Applications. *Book*, 2001.
- [12] **I. Sutskever, J. Martens, G. Hinton.** Generating Text with Recurrent Neural Networks. *28th International Conference on Machine Learning*, pages 1017 - 1024, 2011.
- [13] **A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin.** Attention is all you need. *31st Conference on Neural Information Processing Systems (NIPS)*, 2017.
- [14] **C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Yanqi Zhou, Wei Li, P. J. Liu.** Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 2019.
- [15] **R. Sennrich, B. Haddow, A. Birch.** Neural Machine Translation of Rare Words with Subword Units. *54th Annual Meeting of the Association for Computational Linguistics, volume 1: Long Papers*, pages 1715–1725, 2016.
- [16] **Xiaodong Liu, Kevin Duh, Liyuan Liu, Jianfeng Gao.** Very Deep Transformers for Neural Machine Translation. *Article from Association for Computational Linguistics (ACL)*, 2020.
- [17] **A. Mohammadshahi, V. Nikoulina, A. Berard, C. Brun, J. Henderson, L. Besacier.** SMaLL-100: Introducing Shallow Multilingual Machine Translation Model for Low-Resource Languages. *Conference on Empirical Methods in Natural Language Processing*, pages 8348–8359, 2022.
- [18] **Sho Takase, Shun Kiyono.** Lessons on Parameter Sharing across Layers in Transformers. *The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP)*, pages 78–90, 2023.
- [19] **A. Aranta, A. Djunaidy, N. Suciati.** Developing an English-to-Indonesian speech-to-text as a foundation for Sasak language translation using the mBART algorithm. *International Conference on Green Energy, Computing and Intelligent Technology (GEn-CITy)*, 2024.
- [20] **A. Smirnov, N. Teslya, N. Shilov, D. Frank, E. Minina, M. Kovacs.** Quantitative Comparison of Translation by Transformers-Based Neural Network Models. *Article from Enterprise Information Systems*, pages 155-174, 2023.

- [21] **A. K. Patel; K. Chakravaty; S. Bilgaiyan; P. S. Mishra; N. Kumari.** English to Hinglish translator using Llama 3. *6th International Conference on Computational Intelligence and Networks (CINE)*, 2024.
- [22] **Jianlin Su, Yu Lu, Shengfeng Pan, A. Murtadha, Bo Wen, Yunfeng Liu.** Enhanced Transformer with Rotary Position Embedding. *Article on arXiv*, 2021.
- [23] **Rui Zhang, Zong-Ben Xu, Guang-Bin Huang, Dianhui Wang.** Global Convergence of Online BP Training With Dynamic Learning Rate. *IEEE Transactions on Neural Networks and Learning Systems*, volume 23, issue 2, pages 330 - 341, 2012.
- [24] **B. R. Bartoldson, A. S. Morcos, A. Barbu, G. Erlebacher.** The Generalization-Stability Tradeoff In Neural Network Pruning. *34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
- [25] **I. Sutskever, O. Vinyals, Quoc V. Le.** Sequence to Sequence Learning with Neural Networks. *28th International Conference on Neural Information Processing Systems*, 2014.
- [26] **S. Bhosale, K. Yee, S. Edunov, M. Auli.** Language Models not just for Pre-training: Fast Online Neural Noisy Channel Modeling. *Fifth Conference on Machine Translation (WMT)*, pages 584–593, 2020.

Приложения

- Большинство вычислений, включая обучение моделей, было проведено на предоставленном НИУ ВШЭ суперкомпьютере “сHARISMa”:
 - Обучение: 1x NVIDIA Tesla V100 32 ГБ NVLink, 1x Intel Xeon Gold 6152 2.1-3.7 ГГц
 - Подсчёт метрик: 4x Intel Xeon Gold 6152 2.1-3.7 ГГц
- Подсчёт метрик на изначальных моделях-трансформерах (T5, mT5, MBart, M2M100, Marian, LLaMA3.2) и рантаймах проходил в Google Colab:
 - 2x Intel Xeon 2.20 ГГц
- Ссылка на используемый датасет: <https://huggingface.co/datasets/aiana94/polynews-parallel>
- Ссылка на код, что запускался в ходе работы и на получившиеся результаты: <https://github.com/DenisKabanov/diploma>
- Ссылка на базовые версии моделей:
 - T5: https://huggingface.co/utrobinmv/t5_translate_en_ru_zh_base_200
 - mT5: <https://huggingface.co/cointegrated/rut5-base-multitask>
 - MBart: <https://huggingface.co/facebook/mbart-large-50-many-to-many-mmt>
 - M2M100: https://huggingface.co/facebook/m2m100_418M
 - Marian: <https://huggingface.co/Helsinki-NLP/opus-mt-ine-ine>
 - LLaMA3.2 - 1B: <https://huggingface.co/Vikhrmodels/Vikhr-Llama-3.2-1B-Instruct>

- Графики дообучения и Latency для оставшихся вариантов прунинга:

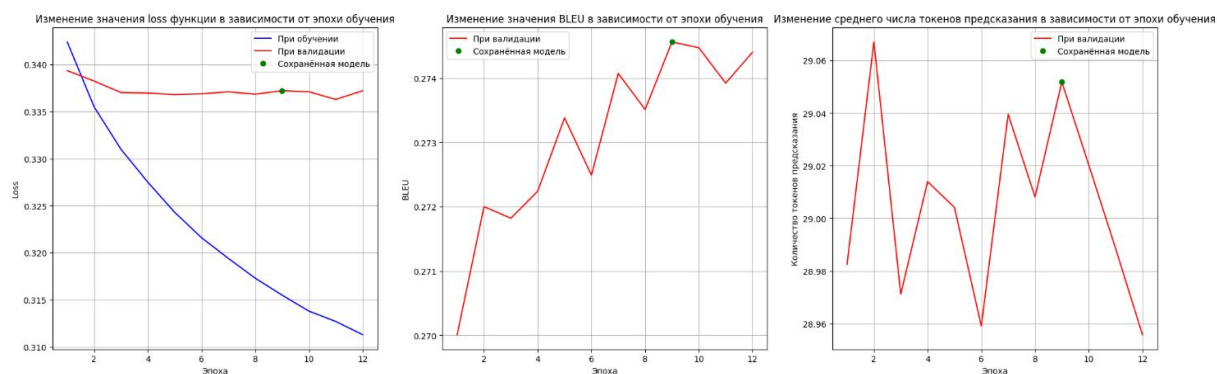


Рис. 13 - Изменение отслеживаемых метрик в процессе обучения для модели T5 с неструктурированным прунингом 33% параметров

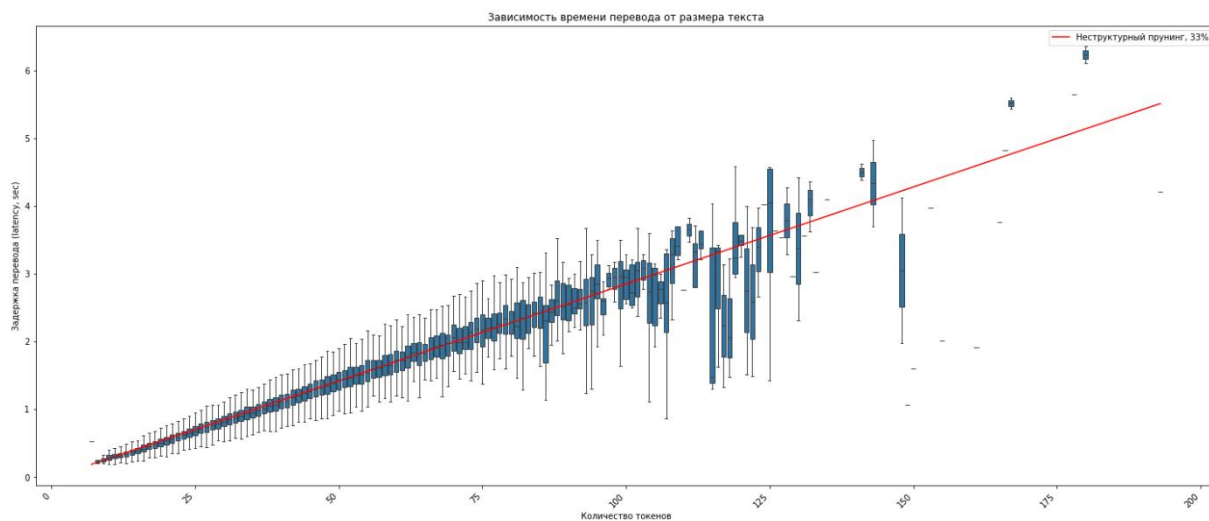


Рис. 14 - Зависимость Latency от размера входа для дообученной модели T5 с неструктурированным прунингом 33% параметров

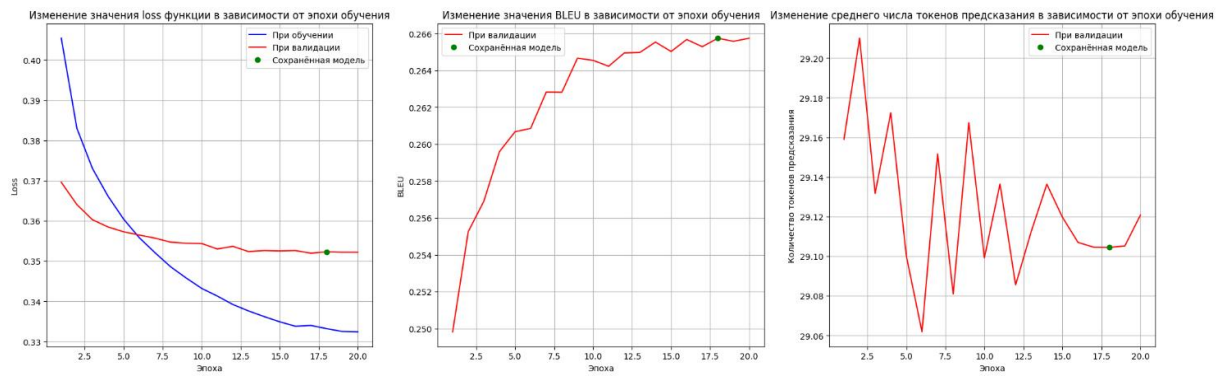


Рис. 15 - Изменение отслеживаемых метрик в процессе обучения для модели T5 с неструктурированным прунингом 50% параметров

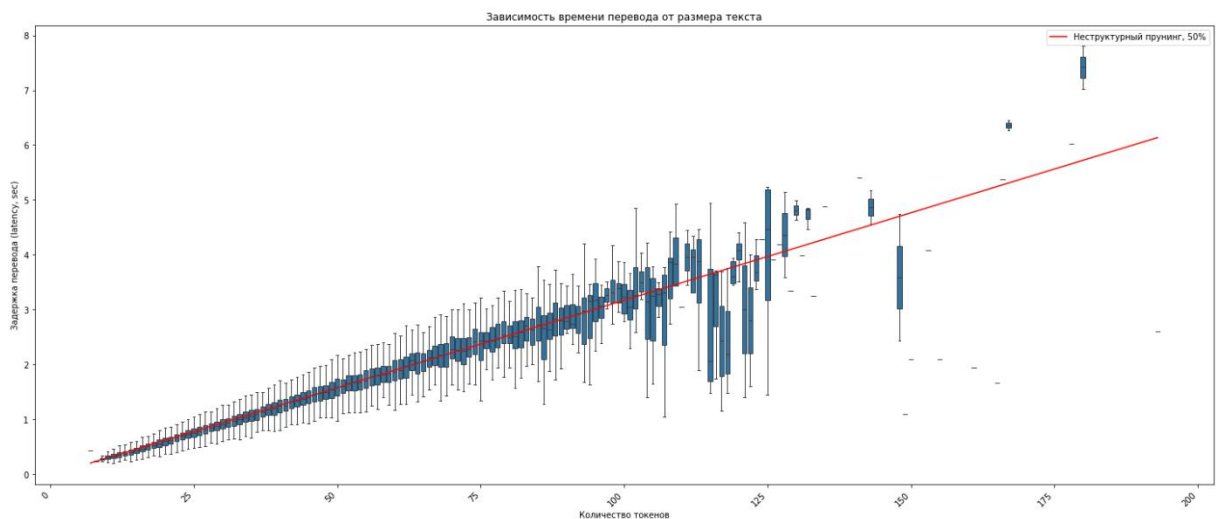


Рис. 16 - Зависимость Latency от размера входа для дообученной модели T5 с неструктурированным прунингом 50% параметров

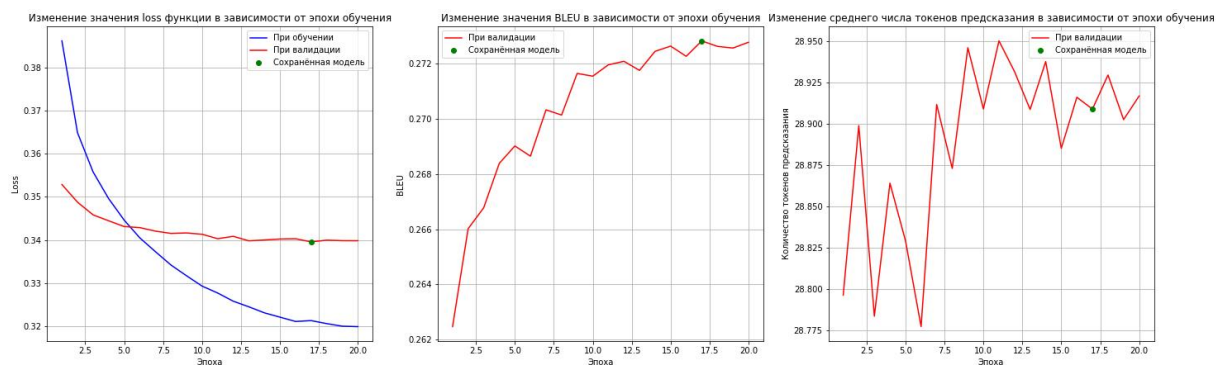


Рис. 17 - Изменение отслеживаемых метрик в процессе обучения для модели T5 с структурированным прунингом 10% параметров

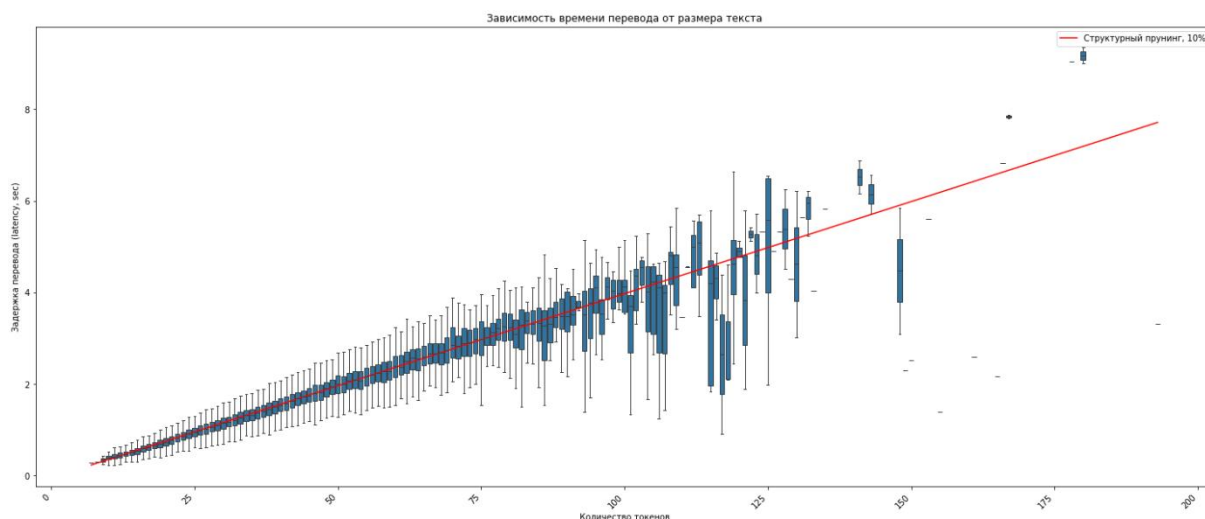


Рис. 18 - Зависимость Latency от размера входа для дообученной модели T5 с структурированным прунингом 10% параметров

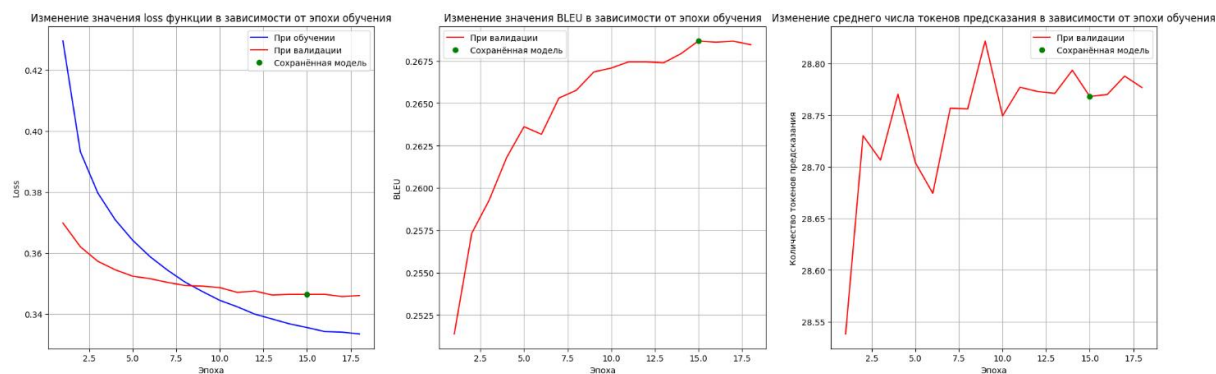


Рис. 19 - Изменение отслеживаемых метрик в процессе обучения для модели T5 с структурированным прунингом 15% параметров

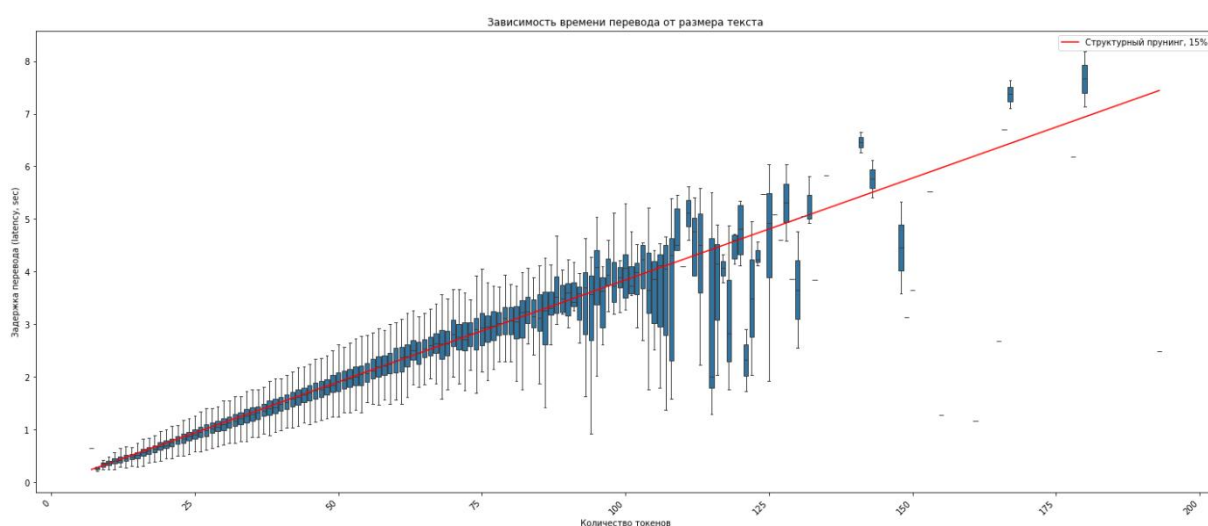


Рис. 20 - Зависимость Latency от размера входа для дообученной модели T5 с структурированным прунингом 15% параметров

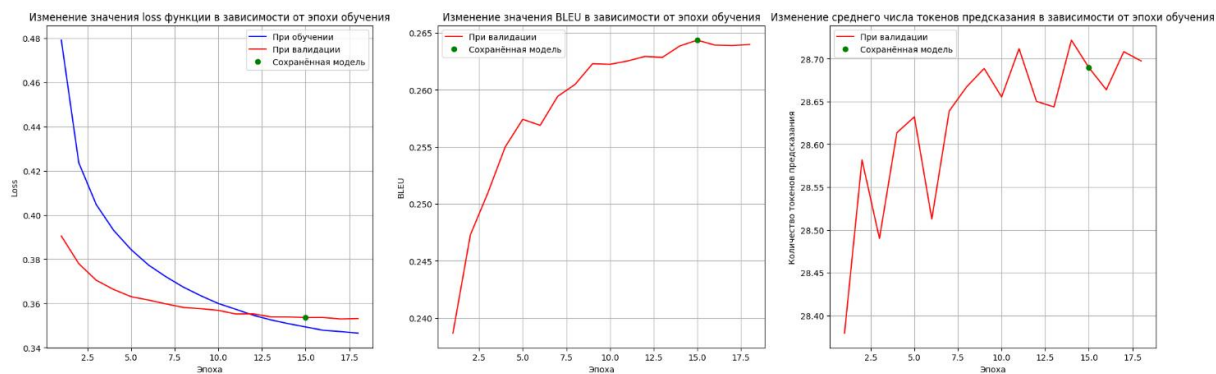


Рис. 21 - Изменение отслеживаемых метрик в процессе обучения для модели T5 с структурированным прунингом 20% параметров

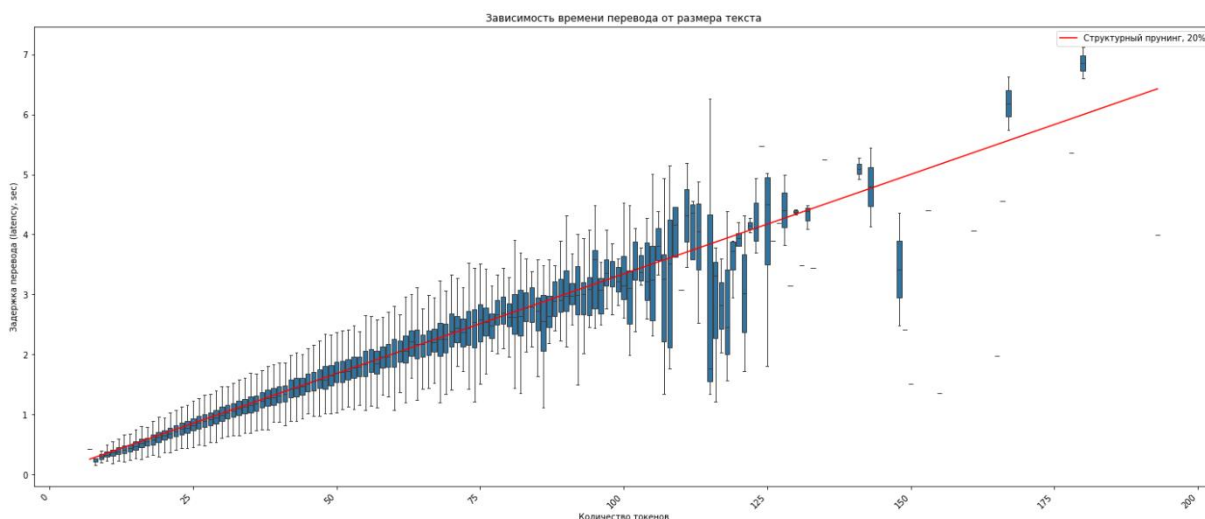


Рис. 22 - Зависимость Latency от размера входа для дообученной модели T5 с структурированным прунингом 20% параметров

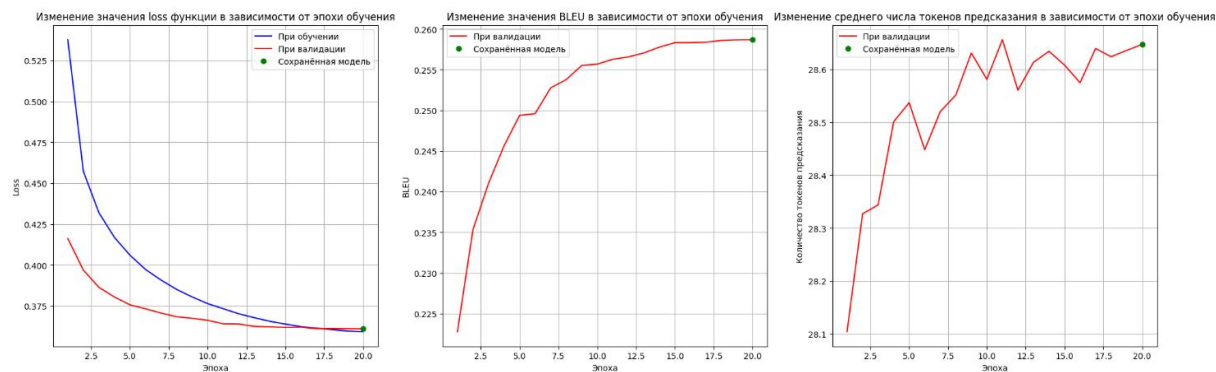


Рис. 23 - Изменение отслеживаемых метрик в процессе обучения для модели T5 с структурированным прунингом 25% параметров

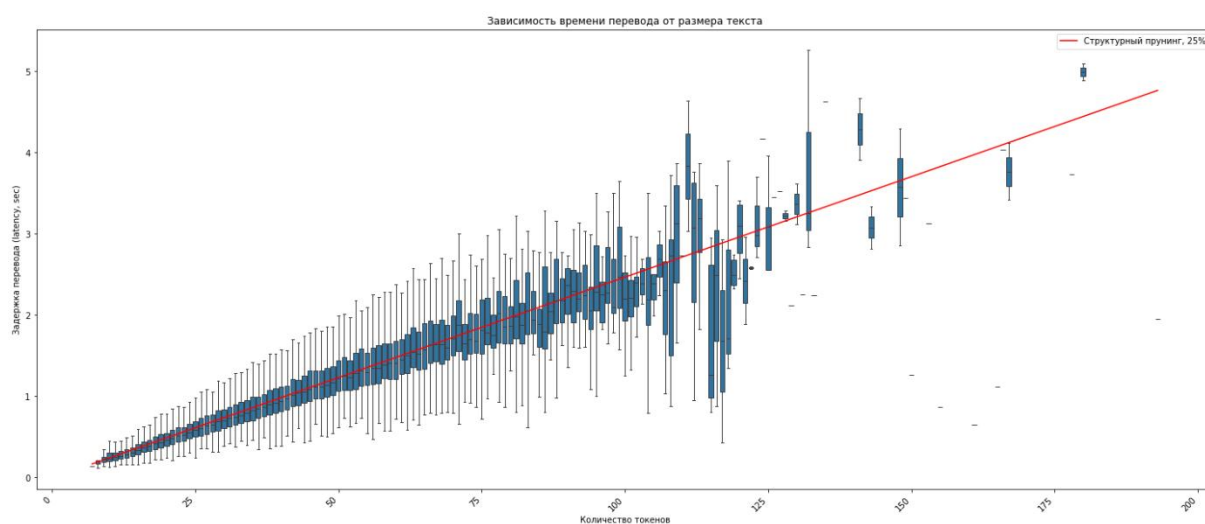


Рис. 24 - Зависимость Latency от размера входа для дообученной модели T5 с структурированным прунингом 25% параметров