Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)

Физтех-школа прикладной математики и информатики
Кафедра интеллектуальной обработки документов

Выпускная квалификационная работа бакалавра

Использование внутренних представлений замороженных трансформерных моделей для решения downstream задач

Автор:
Студент Б05-031 группы
Дремов Александр Олегович

Научный руководитель:
Воропаев Павел Михайлович

Москва 2024

Аннотация

Использование внутренних представлений замороженных трансформерных моделей для решения downstream задач
Дремов Александр Олегович

Большие трансформерные модели сегодня являются одним из основных способов решения различных задач. Например, они используются для суммаризации текста, распознавания речи, определения сентимента текста и т. д.

Однако, большие модели требуют значительных ресурсов, что создаёт проблемы при необходимости решать сразу несколько задач. В этой работе предлагается решение проблемы, согласно которому уже существующая трансформерная модель, решающая основную задачу (базовая модель), может быть переиспользована для решения смежных задач над теми же входными данными. Предложенное решение эффективно переиспользует посчитанные результаты скрытых слоев базовой модели для решения смежных задач без внесения изменений в базовую модель.

По результатам экспериментов, предложенный подход получает качество, сопоставимое с SOTA (state-of-the-art) решениями в области, конкурируя с решениями, использующими полное дообучение модели.

Содержание

1 Введение

1.1 Информация о сфере задачи

Трансформеры, впервые представленные в статье «Attention Is All You Need» [1] командой Google в 2017 году, продемонстрировали выдающиеся результаты по сравнению с рекуррентными и сверточными нейросетями. Они стали основой для создания таких моделей, как BERT («Pre-training of Deep Bidirectional Transformers for Language Understanding» [2]), GPT («Language Models are Unsupervised Multitask Learners» [3]) и T5 («Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer» [4]), которые кардинально изменили подходы к решению сложных задач обработки естественного языка (natural language processing, NLP). Основная компонента этих моделей — механизм внимания, который позволяет эффективно обрабатывать последовательности данных.

Механизм внимания работает по принципу «выделения» ключевых элементов входной последовательности, на которых модель должна сфокусироваться для выполнения задачи. Это позволяет трансформерным моделям учитывать контекст и отношения между элементами независимо от их положения в последовательности. Это решает характерную для рекуррентных нейронных сетей проблему плохой работы с длинными зависимостями.

Трансформерные модели нашли огромное применение в автоматической обработке текста, включая задачи машинного перевода, суммаризации текста, анализа тональности, ответа на вопросы и генерации текста. Например, модель GPT-3, позволяет не только вести осмысленные диалоги с пользователями, но и выполнять глубокий анализ текстов для поиска и извлечения информации. В коммерческих приложениях подобные модели используются в системах рекомендаций, чат-ботах для службы поддержки и переводчиках.

Однако стоит отметить, что трансформерные модели требовательны к ресурсам. Так, современные модели, которые демонстрируют наилучшие результаты (state-of-the-art модели), обычно очень большие. Например, модель GPT-3 имеет более 175 миллиардов параметров. Это создаёт определённые сложности при их использовании, особенно на устройствах с ограниченными ресурсами.

Поэтому актуальность исследования обусловлена широким спектром применений трансформерных моделей (трансформеры) в различных задачах NLP. Таким образом, необходимо разрабатывать более эффективные и ресурсоёмкие методы использования трансформерных моделей.

1.2 Архитектура трансформерной модели

Классическая трансформерная модель состоит из энкодера и декодера, которые содержат слои, состоящие из multihead-attention [1], нормализации и полносвязных слоев. Визуализация архитектуры представлена на рис. 1.

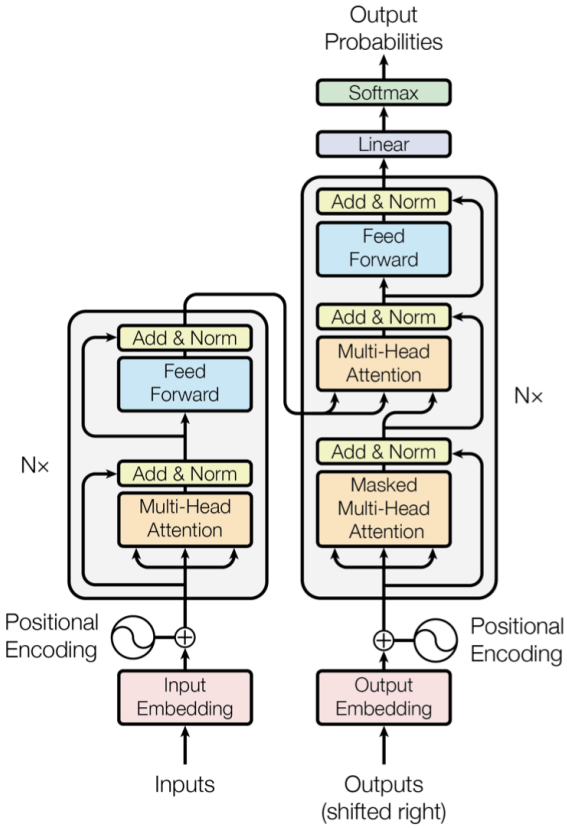


Рис. 1: Визуализация классической архитектуры трансформерной модели из статьи «Attention Is All You Need» [1]. Слева — блоки энкодера, справа блоки декодера.

В корне архитектуры лежит механизм внимания (dot-product attention) [1]. Для тройки тензоров Q, K, V алгоритм записывается так:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

где d_k — размерность признаков Q и K. Тензоры Q, K, V — входные данные алгоритма. В трансформерном слое используется алгоритм multi-head attention:

$$\begin{aligned}\text{MultiHead}(Q, K, V) &= \text{Concat}\left(head_1, \ldots, head_h\right)W^O,\\ head_i &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right),\end{aligned}$$

где W_i^Q, W_i^K, W_i^V, W^O — обучаемые матрицы. В энкодерных слоях и в masked attention слоях декодера, $Q = K = V$ — такая постановка называется «self-attention». В декодерных слоях K, V представляют выходы энкодера, а Q — это выходы предыдущей операции декодерного слоя.

Резюмируя, энкодер состоит из N идентичных слоев, каждый из которых включает:

1. Слой multi-head attention.
2. Слой полносвязной нейронной сети (feed-forward network, FFN).
3. Нормализацию (Layer Normalization).

4. Резидуальные соединения (residual connections).

Аналогично, декодер также состоит из N идентичных слоев, каждый из которых включает:

1. Слой masked multi-head attention, предотвращающий внимание на будущие позиции при генерации последовательности.
2. Слой multi-head attention, аналогичный энкодерному, который использует в качестве K, V выходы энкодера.
3. Слой полносвязной нейронной сети (feed-forward network, FFN).
4. Нормализацию (Layer Normalization).
5. Резидуальные соединения (residual connections).

Внутренние слои трансформера называют скрытыми слоями, а данные на их выходе — скрытыми состояниями. Финальный слой декодера преобразует выходные представления в прогнозируемое распределение вероятностей.

2 Постановка задачи

Важно отметить, что часто помимо решения основной NLP задачи имеется несколько смежных задач для одних и тех же входных данных. Например:

- суммаризация текста и поиск именованных сущностей в нём,
- перевод текста и его классификация,
- распознавание текста по аудиозаписи голоса и классификация эмоции пользователя.

Причём одна из задач считается основной (главной), на которую идут почти все вычислительные ресурсы для получения высокого качества. Основная задача — это та, которая осуществляет ключевой функционал продукта и должна иметь, насколько возможно, высокое качество.

Выходит, что смежная задача вынуждена работать в условиях ограниченных вычислительных мощностей. В такой постановке не представляется возможным использование отдельных больших state-of-the-art моделей для решения смежных задач ввиду высокой требовательности к ресурсам, упомянутой ранее.

С другой стороны, большое количество независимых больших моделей для решения каждой отдельной задачи требует последовательную обработку ввиду требовательности к ресурсам. Такой подход замедляет работу всего сервиса и увеличивает потенциальное время ответа.

В случае проведения вычислений на пользовательских устройствах использование набора из больших трансформерных моделей для разных задач просто не представляется возможным ввиду сильных ограничений на доступные ресурсы.

Таким образом, задачу работы можно сформулировать так: без негативного влияния на качество модели, решающей главную задачу, необходимо эффективно решать смежные задачи в условиях ограниченных вычислительных мощностей.

3 Обзор существующих решений

3.1 Общие решения

Для экономии ресурсов при использовании трансформерных моделей были предложены различные методы, среди которых можно выделить основные:

1. Квантизация (Quantization)

   Квантизация — это общий подход понижения точности вычислений для их ускорения с, быть может, просадками в качестве.

   Так, например, в «Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference» [5] Google предлагает метод модификации модели для вычисления предсказаний в более быстрой целочисленной арифметике, а не в числах с плавающей точкой.

2. Дистилляция знаний (Knowledge Distillation)

   Основная идея дистилляции заключается в том, что компактная модель обучается на выходных данных крупной модели, которые могут содержать более богатую и полезную информацию по сравнению с обычной разметкой данных. Это позволяет упростить модель, сохраняя при этом качество.

   Например, DistilBERT [6] является компактной альтернативой BERT со схожим качеством.

Такие методы действительно экономят ресурсы. Однако, при наличии основной задачи, часто полученный прирост в скорости обменивается на увеличение размера модели для повышения качества. Поэтому, при наличии смежных задач, проблема ограниченных ресурсов по-прежнему остаётся, даже при применении указанных выше подходов.

3.2 Решения с использованием одной модели

Для решения нескольких задач с использованием одной трансформерной модели была предложена серия подходов:

1. Обучение на множество задач (Multi-task learning)

Данный подход предполагает обучение одной модели для выполнения нескольких задач одновременно, что позволяет снижать вычислительные затраты путем совместного использования общих представлений и параметров. Примером является модель T5 [4], которая использует унифицированный подход к обработке различных NLP задач.

Однако, такой подход не всегда применим по нескольким причинам: обучение с нуля большой модели для решения дополнительной задачи может быть слишком затратно. Также, необходимо контролировать баланс данных разных задач, чтобы не переобучаться на одной из них, что может быть сложно.

Помимо этого, в таком подходе одна модель обычно решает одну задачу за один проход. Тогда, для решения нескольких задач, требуется несколько проходов, что негативно влияет на производительность системы.

2. Разделение на домены (Domain-specific fine-tuning)

   Этот метод включает адаптацию модели на основе данных из разных доменов и задач. Здесь трансформерная модель может быть обучена на данных одного домена и затем дообучена на другом, при этом предполагается, что важные знания сохраняются и переносятся между задачами.

   Проблемой является то, что при дообучении на новую задачу качество на исходной может начать проседать, что недопустимо в рассматриваемой постановке. Данная проблема описывается в литературе как проблема забывания или «Forgetting Problem» [7].

3. Синергетическое использование (Cross-task adapters)

   В этом подходе используются адаптерные модули, оптимизированные для многозадачности. Они позволяют одной основной модели эффективнее выполнять несколько различных задач. Этот метод может включать динамическое внедрение и обучение новых слоев адаптеров, специфичных для каждой задачи. Адаптер — это небольшая нейронная сеть, которая встраивается в архитектуру слоев трансформера и модифицирует внутренние вычисления.

   Например, в статье «Cross-Lingual Transfer with Target Language-Ready Task Adapters» [8] рассматривается возможность обучения адаптерных слоев для адаптации задачи под разные языки.

   Проблемой данного подхода в случае смежных задач является необходимость полного перерасчета базовой модели. Ввиду того, что адаптеры встраиваются в архитектуру модели, вычисления для разных задач будут совершенно разными.

4. Методы эффективного дообучения (PEFT).

   Высокую популярность в сфере дообучения трансформерных моделей получили методы эффективного дообучения (parameter efficient tuning, PEFT). В статье

«Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning» [9] приведен обзор самых популярных методов.

Авторы разделяют подходы на несколько категорий, диаграмма разделения представлена на рис. 2. Однако, при изучении каждой категории можно сделать вывод, что многие методы PEFT не подходят для решения поставленной задачи. Так, аддитивные методы [10, 11], селективные методы [12, 13, 14] и методы, основанные на репараметризации [15, 16], меняют исходную модель, что противоречит постановке задачи.

Задачу потенциально могут решить soft-prompts [17] — методы модификации входа трансформерной модели и ее внутренних состояний для адаптации модели под новую задачу. Однако данный метод не способен сильно изменить поведение модели, а также изменяет вычисления, что требует многократного применения модели в случае нескольких задач.

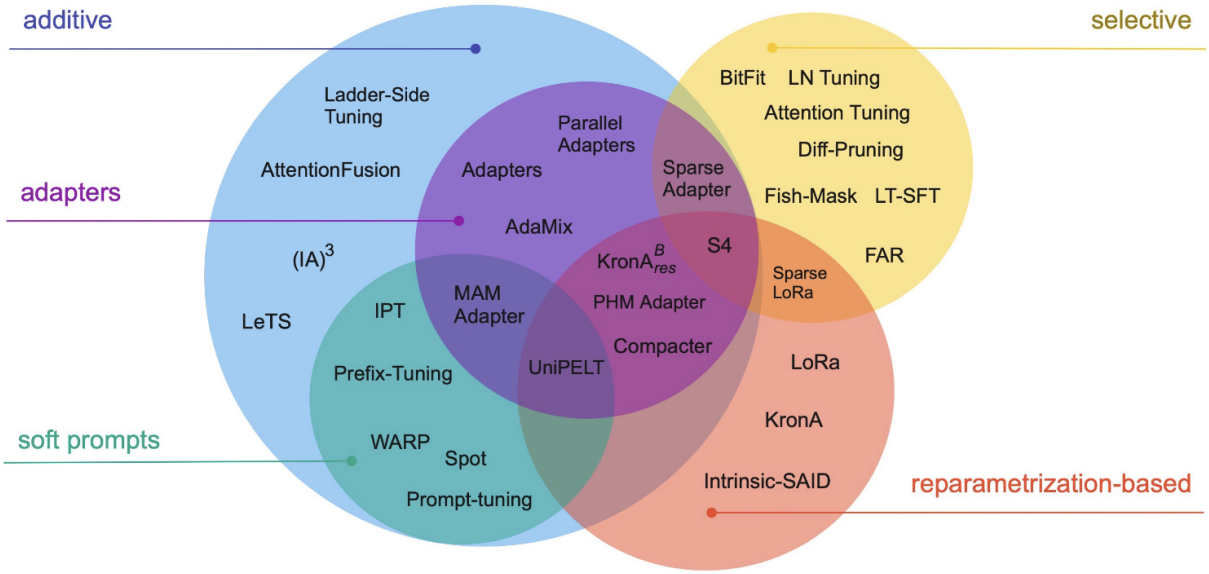


Рис. 2: Разбиение алгоритмов PEFT на категории. «Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning» [9]

Таким образом, существуют методы для экономии ресурсов при решении смежных задач над одними входными данными. Однако популярные методы рассматривают смежные задачи как равноценные и не фокусируются на случае основной задачи и смежной. Это различие важно, так как:

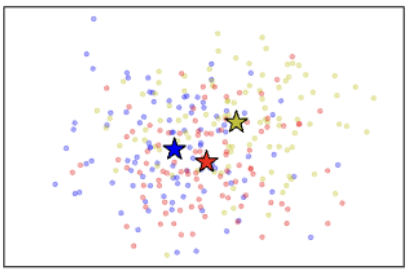
- смежная задача вынуждена работать в условиях малого числа ресурсов,
- добавление смежной задачи не должно негативно влиять на качество выполнения основной задачи.

3.3 Решения с использованием одной модели без модификаций

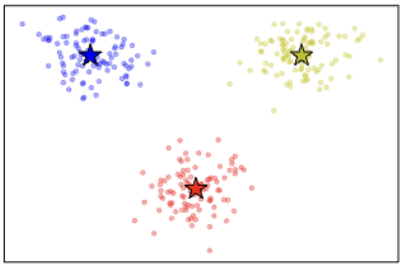
Помимо литературы, рассмотренной выше, существуют подходы для решения смежных задач, основанные на дополнении существующей модели без какого-либо ее изменения.

1. «Hidden State Variability of Pretrained Language Models Can Guide Computation Reduction for Transfer Learning» [18]

   В данной статье рассматривается подход для решения задачи классификации. Суть метода заключается в использовании предобученной модели трансформера, из которой выбирается «наиболее информативный» скрытый слой по метрике вариативности выходов. Информативным будет считаться слой, в котором межклассовый разброс большой, а внутриклассовый разброс маленький. Примеры представлены на рис. 3.



(а) Высокая внутриклассовая вариативность, низкая межклассовая вариативность



(б) Низкая внутриклассовая вариативность, высокая межклассовая вариативность

Рис. 3: Пример из «Hidden State Variability of Pretrained Language Models Can Guide Computation Reduction for Transfer Learning» [18]. На рисунках изображены выходные значения различных скрытых слоев модели. Цветом обозначен истинный класс объекта, породившего изображенные значения. Слева (3а): малоинформативный для задачи слой. Справа (3б): высокоинформативный для задачи слой

   Метод интересен тем, что он не требует изменений базовой модели и эффективно переиспользует ее расчеты. Однако авторы замечают, что метод выбора слоя не всегда может работать и является лишь эвристикой. Тривиальным примером, когда алгоритм не справляется, является расположение кластеров в форме концентрических кругов. Такая структура легко разделима на классы, но не соответствует эвристике авторов.

   Помимо этого, алгоритм выбора слоя заточен под задачу классификации, и случай работы с другими задачами не разбирается.

2. «An Algorithm for Routing Vectors in Sequences» [19]

   Решение, предложенное в статье [19] использует идею из капсульных нейронных сетей [20] для комбинирования всех скрытых состояний трансформерной модели.

Алгоритм динамически выбирает полезные признаки из каждого слоя модели и использует их для вычислния финального предсказания. Однако, алгоритм требует хранения всех скрытых состояний базовой модели, что для больших трансформерных архитектур может достигать нескольких гигабайтов памяти на один входной пример.

Недостатком алгоритма на практике является введение нового типа слоя с нетривиальными тензорными операциями. Для быстрого применения данного алгоритма необходимо эффективно реализовать предложенные операции методами видеокарты, что не тривиально.

На рис. 4 представлен пример работы алгоритма на одном примере. Видно, как для разных токенов алгоритм использует различные скрытые слои нейронной сети для получения предсказания.

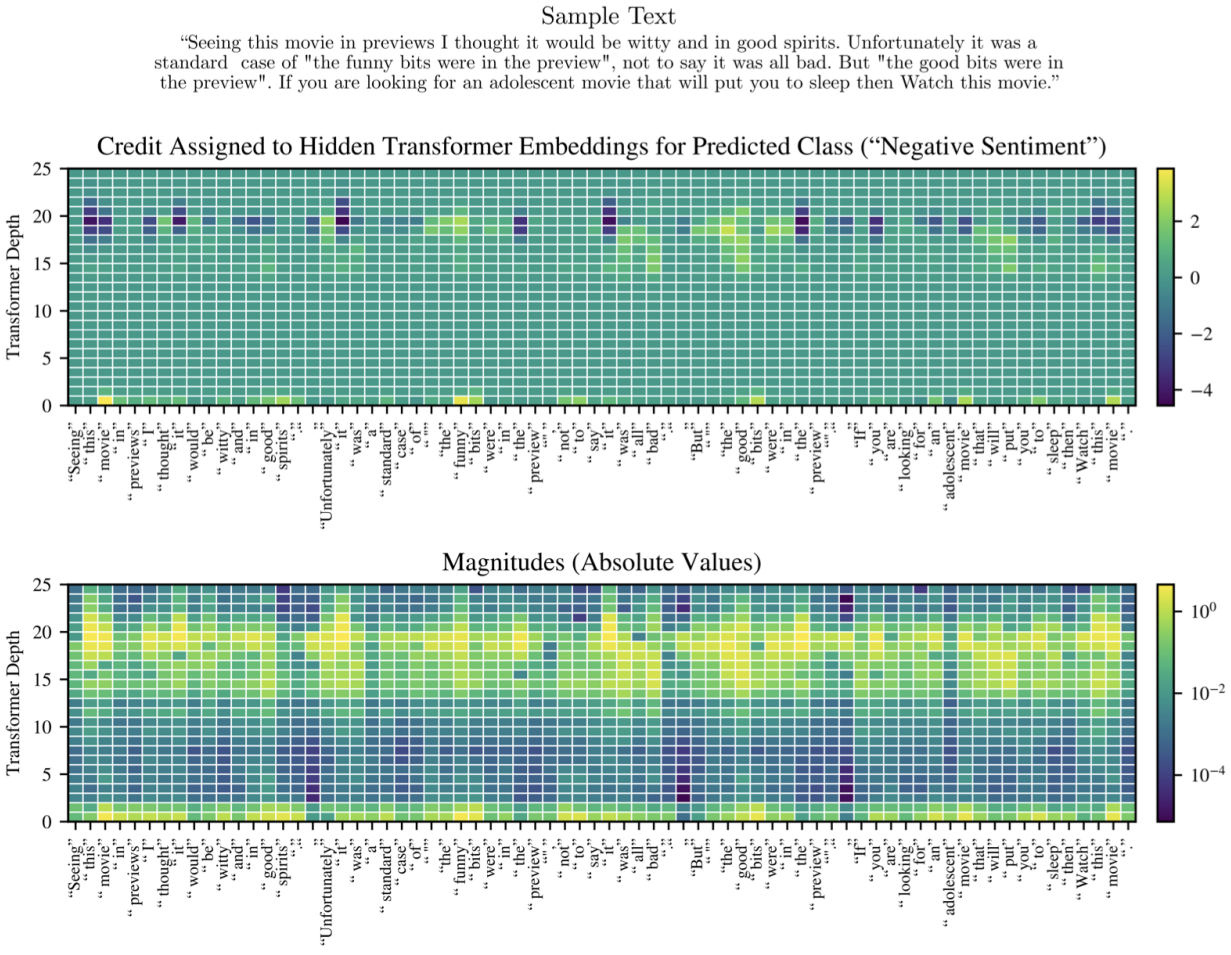


Рис. 4: Визуализация алгоритма из статьи «An Algorithm for Routing Vectors in Sequences» [19]. Показана величина вклада каждого скрытого слоя модели в предсказание. Величина влияния слоя вычислена агрегацией влияния по всем признакам слоя

3. «Head2Toe: Utilizing Intermediate Representations for Better Transfer Learning» [21]

   Данный метод, разработанный Google, предлагает брать все скрытые слои и обу-

чать на их конкатенации необходимую модель. Пусть L — число скрытых слоев трансформерной модели, а h_i, $i \in [1, L]$ — скрытые состояния, тогда предложенный алгоритм представим так:

$$h_{all} = [a_1(h_1), a_2(h_2), \ldots, a_L(h_L)],$$
$$z'_L = h_{all} W_{all},$$

где a_i, $i \in [1, L]$ — понижение размерности усреднением (average pooling) и нормализация, W_{all} — обучаемая матрица, а z'_L — тензор признаков, используемый для финального предсказания.

Полученная конструкция обучается сквозным образом (end-to-end) на необходимую задачу с group-lasso регуляризацией матрицы W_{all}. Благодаря регуляризации, матрица W_{all} получается разреженной, и из нее далее выделяют подмножество строк, обладающих наибольшей «важностью». Это позволяет уменьшить количество признаков в h_{all}. Визуализация алгоритма представлена на рис. 5.

Аналогично предыдущей статье, базовая модель, используемая для алгоритма, остается неизменной. Алгоритм показывает хорошие результаты на тестовых наборах данных и решает поставленную задачу.

Однако алгоритм не лишен недостатков — главная проблема это необходимость сохранения выходов всех слоев для расчета h_{all}, что требует больших объемов памяти, как замечают сами авторы.

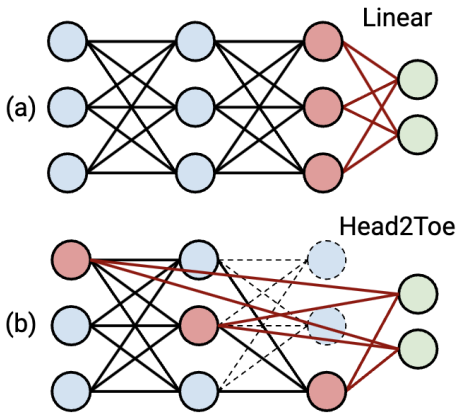


Рис. 5: Визуализация алгоритма Head2Toe [21]. Показана ключевая особенность алгоритма — выбор различных признаков из скрытых слоев модели для получения финального предсказания

Эту проблему частично решает регуляризация W_{all} и выбор доли признаков. Однако оптимальная доля признаков, получаемая алгоритмом, часто достаточно велика, как следует из приложения к работе авторов. В случае больших трансформерных моделей, хранение всех скрытых состояний может достигать гигабайтов дополнительной памяти для одного примера.

4 Исследование и построение решения задачи

4.1 Цель работы

Разработать алгоритм, оперирующий в условиях нехватки ресурсов и в присутствии некоторой большой трансформерной модели, решающей главную задачу. Без негативного влияния на качество основной модели, алгоритм должен эффективно решать смежные задачи над теми же входными данными.

4.2 Задачи работы

1. Подобрать условия для разработки алгоритма — выбрать базовую модель, и оценочные наборы данных (benchmarks).
2. На основании литературы и существующих решений, определить качество сторонних решений на оценочных наборах данных.
3. Разработать алгоритм, переиспользующий вычисления основной модели для решения смежных задач.
4. Обучить полученный алгоритм на оценочных наборах данных и оценить качество на отложенном наборе данных (тестовый набор данных).
5. Оценить качество решения задачи/задач с помощью простых моделей (baseline).
6. Сформулировать выводы, заключение о работоспособности алгоритма.
7. Сформулировать вопросы, требующие дополнительного изучения.

4.3 Выбор условий разработки и оценки качества

В качестве базовой модели было решено выбрать две модели: RoBERTa-Large [22] и BART [23]. RoBERTa-Large — это модель, основанная на классической архитектуре BERT [1], которая была усовершенствована и доработана. Предобученная модель решает задачу MLM (masked language modelling) — предсказывание маскированных токенов в тексте. Данный выбор обусловлен универсальностью модели и её несмещённостью в какую-либо специфичную задачу, что делает последующее сравнение на различных оценочных данных более честным.

Помимо RoBERTa-Large, для экспериментов выбрана модель BART, предобученная на задачу суммаризации текста. BART — это энкодер-декодер модель, в которой энкодер соответствует архитектуре BERT, а декодер имеет авторегрессионную архитектуру, схожую с GPT. Данная конструкция обучается сквозным образом на задачу суммаризации. Данная модель, в отличие от RoBERTa-Large, узко специализирована, что поможет оценить качество алгоритма в постановке, близкой к реальным условиям.

В качестве оценочных наборов данных были выбраны распространенные наборы для оценки качества модели:

1. **IMDB Sentiment** [24].

   Данный набор данных используется для анализа тональности текста, конкретно — в контексте кинообзоров. Цель заключается в том, чтобы классифицировать обзоры фильмов на положительные или отрицательные.

2. **GLUE CoLA** [25].

   Датасет GLUE CoLA (Corpus of Linguistic Acceptability) направлен на оценку способности различать грамматически правильные и неправильные предложения. CoLA входит в состав бенчмарка GLUE (General Language Understanding Evaluation), который используется для комплексной оценки качества различных NLP моделей.

3. **CoNLL** [26] NER.

   Набор данных CoNLL используется для оценки качества именованного распознавания сущностей (NER). Так, CoNLL-2003 содержит данные на английском языке с метками для различных сущностей: имена людей, организаций, мест, общее. Каждому входному слову из текста необходимо присвоить ровно один из перечисленных классов.

Данный выбор данных обусловлен желанием проверить алгоритм на задачах различной сложности и формата: от простой классификации последовательности (IMDB) до классификации на уровне токенов (NER). Таким образом, по полученным результатам о качестве алгоритма можно будет судить в полной мере. Для понимания того, насколько достигаемое качество соотносится с мировыми лучшими решениями, часть лучших результатов выписана в приложении 6.4.

4.4 Мотивация и планирование алгоритма

Из постановки задачи известно, что имеется некоторая большая (основная) трансформерная модель, решающая главную задачу. Идея предложенного подхода состоит в переиспользовании вычислений основной модели без её модификации. Очевидно, качество решения главной задачи останется неизменным из-за неизменности основной модели. Тогда, если алгоритм сможет получать информативные признаки из уже посчитанных выходов основной модели, то можно будет качественно решать смежные задачи. Причём такое решение будет вычислительно простым, если обработка полученных от основной модели признаков будет эффективной и лёгкой.

Для получения информативных признаков решено использовать выходные данные скрытых слоёв трансформерной модели. Основой данной идеи являются статьи [27, 28, 29], указывающие на факт, что различные скрытые слои выделяют признаки различного уровня сложности и абстракции. В том числе, на потенциальный успех данного подхода указывает рассмотренная ранее статья [21], в которой авторы предлагают алгоритм переиспользования результатов скрытых слоёв для решения смежных задач.

Таким образом, предлагается разработать алгоритм, эффективно извлекающий полезную для смежной задачи информацию из уже посчитанных признаков основной модели. При таком формате решения можно будет легко и быстро обучаться на решение любого набора смежных задач при минимальной трате ресурсов на момент применения алгоритма.

5 Описание практической части

5.1 Описание алгоритма

Руководствуясь идеей, что скрытые состояния трансформерной модели выделяют признаки разного уровня, построим следующий алгоритм. Пусть некоторая трансформерная модель, состоящая из l слоев, получает на вход данные X размерности (B, T, K) — размерность числа примеров (batch), размерность последовательности и размерность признаков соответственно. В постановке работы с текстом, K соответствует размерности embedding-слоя модели. Тогда, основная модель в процессе использования вычисляет выходы каждого из l слоев:

$$Z_i,\ i \in [1, l] \text{ — номер скрытого состояния.}$$

Применим следующее преобразование к каждому из слоев и получим новые признаки:

$$f_i = N(a(Z_i)W_i^T),\ i \in [1, l],$$

где a — некоторая нелинейность (функция активации, например GELU [30]), $\underset{k\times K}{W_i}$ — обучаемый параметр линейного преобразования признаков базовой модели в меньшее пространство размерности k, N — нормирование по размерности признаков (element-wise affine layer norm):

$$N(x) = \frac{x - \underset{dim=last}{\mathbb{E}} x}{\sqrt{\underset{dim=last}{\mathbb{D}} x + \varepsilon}}.$$

Идея такого преобразования в том, что обучаемое линейное отображение должно выделить полезную для задачи информацию из каждого конкретного скрытого слоя. Нормировка приводит признаки с разных слоев к единому виду.

Однако, потенциально не каждый скрытый слой имеет полезные для задачи признаки. Чтобы дать модели способность выбрать полезные слои, введем параметр $\underset{l\times 1}{p}$ — вектор, компоненты которого определяют полезность каждого скрытого слоя для задачи.

Используя введенный параметр, можно объединить признаки с каждого скрытого слоя, предварительно отнормировав p с помощью softmax для получения распределения на слоях:

$$\text{Softmax}(p_i) = \frac{\exp(p_i)}{\sum_j \exp(p_j)},$$
$$\tilde{p} = \text{Softmax}(p)$$
$$F = \sum_{i=1}^{l} \tilde{p}_i \cdot f_i.$$

Тогда итоговый алгоритм получения признаков F для решения необходимой задачи можно записать в матричной форме так:

$$\tilde{p} = \text{Softmax}(p),$$
$$F = \left(\underset{dim=last}{\text{Stack}} \left[N\left(a\left(Z_i\right) W_i^T\right) \mid i \in [1, l] \right] \times \tilde{p} \right) \in \mathbb{R}^{B \times T \times k}.$$

Кратко резюмируя алгоритм, идея состоит в отображении скрытых состояний в некоторое меньшее, общее для всех слоев пространство и взвешенном суммировании полученных признаков с весами, отражающими «важность» каждого слоя для задачи. Данные веса являются обучаемыми, что дает модели возможность выбрать полезные для задачи слои.

Над полученными признаками F далее можно строить произвольную нейросеть для решения необходимой задачи. Визуализация алгоритма представлена на рис. 6.

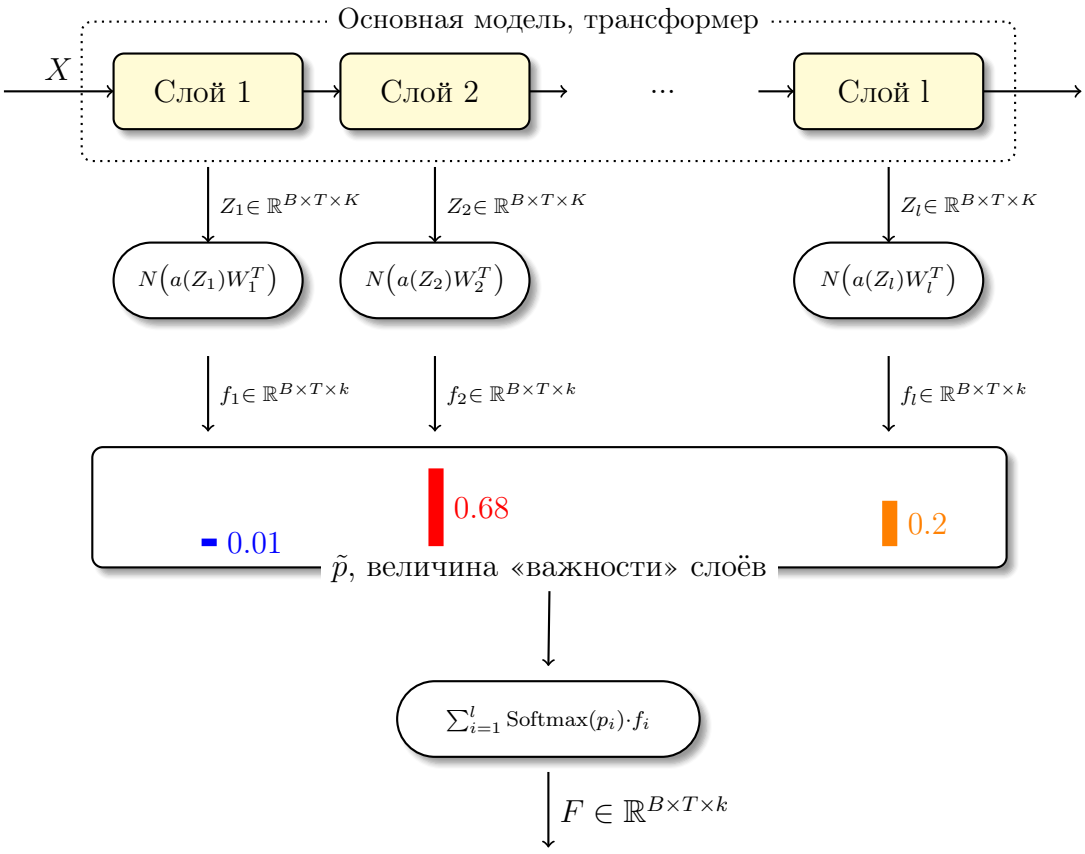


Рис. 6: Визуализация предложенного алгоритма

5.2 Адаптация под конкретные задачи

Полученные признаки можно использовать далее для обучения на необходимую задачу сквозным образом (end-to-end). Очевидно, что над признаками F можно применять любые алгоритмы для работы с последовательностями признаков. В зависимости от задачи, в данной работе над тензором F выполнялись следующие преобразования.

1. **Классификация последовательности**

   В данной задаче для признаков F применяется алгоритм self-attention [1]. Далее, выполняется агрегация усреднением по размерности последовательности $(B, T, k) \to (B, k)$ и дальнейшее линейное проецирование в пространство логитов (B, n_{classes}), где n_{classes} — количество финальных классов.

2. **Классификация токенов (NER)**

   Классификация токенов потребовала добавления более мощной нейронной сети над признаками F. Однако, предложенное преобразование все еще мало в сравнении с основной моделью.

   Так, были использованы несколько классических энкодерных слоёв [1], описанных в введении данной работы. Изменено было лишь расположение нормализации — более актуальные статьи [31] указывают, что расположение нормализации до attention-блоков ведет к более стабильному обучению.

   Полученные признаки (B, T, k) проецируются линейным преобразованием в пространство логитов для каждого из входных токенов: $(B, T, n_{\text{classes}})$, где n_{classes} — количество классов.

5.3 Замечание об эффективном применении алгоритма

Легко заметить, что в силу того, что спроецированные состояния f_i взвешено суммируются, при применении алгоритма на практике достаточно хранить лишь один дополнительный тензор размера (B, T, k). Тогда, можно обновлять хранимое значение, добавляя к нему новый f_i с соответствующим коэффициентом. Таким образом, алгоритм требует совсем немного дополнительной памяти при применении.

Если допустить наличие не одной, а n дополнительных задач, то расширение алгоритма для каждой из них потребует в сумме тензор размера $n \times (B, T, k)$. Такое масштабирование достаточно эффективно, что позволяет применять алгоритм сразу для набора смежных задач.

Помимо этого, в практической части будет показано, что компонента важности для весомой части слоев обнуляется. Это значит, что при применении алгоритма не нужно будет агрегировать все скрытые состояния, что делает вычисления еще более эффективными.

Отметим отдельно, что базовая модель остается неизменной, а добавляемое предложенным алгоритмом число параметров и операций мало в сравнении с базовой моделью, что делает применение алгоритма на практике эффективным.

5.4 Обучение и результаты

5.4.1 Постановка обучения

Обучение моделей проводилось с помощью библиотеки PyTorch [32] на видеокартах NVIDIA. Оптимизационный алгоритм для всех обучений — градиентный спуск в модификации Adam [33].

Так как каждая из решаемых задач является задачей классификации, оптимизируемый функционал — кросс-энтропия:

$$H(p, q) = -\sum_x p(x) \log q(x).$$

В качестве регуляризаций применялся weight decay и dropout [34]. Величина шага оптимизации (learning rate) уменьшалась в два раза при отсутствии улучшений в течение некоторого периода. Обучение прерывалось при отсутствии улучшений длительное время.

Для увеличения способности модели менять параметр p в ходе оптимизации быстрее, в алгоритме вместо p используется $\alpha \cdot p$, где α — гипер-параметр. Также применялась регуляризация на p: в оптимизируемый функционал добавлялась энтропия распределения на скрытых слоях с некоторым коэффициентом λ. Такая регуляризация должна двигать оптимизацию в сторону выделения лишь действительно важных слоев, а не оставаться у распределения близкого к равномерному.

В ходе обучения также было замечено, что изменением параметра p практически невозможно добиться полного «зануления» малоинформативных слоев — некоторые компоненты важности близки к нулю, но не равны ему. Выходит, что почти исключенные слои не добавляют полезной информации, но могут быть использованы моделью для переобучения под тренировочные данные. Поэтому, в ходе обучения пробовался метод обнуления компоненты важности для малоинформативных слоев по некоторому порогу (distribution cutoff) и последующая перенормировка распределения $\tilde{p}$.

Помимо этого, было замечено улучшение качества на некоторых задачах при добавлении позиционных эмбеддингов к тензору F — объяснить это можно тем, что в такой модификации последующим слоям не нужно выделять позиционную информацию из существующих признаков, что позволяет модели сфокусироваться на решении задачи и эффективнее использовать параметры.

Во всех экспериментах базовой моделью являлась RoBERTa-large или BART — их веса были заморожены и не обучались.

5.4.2 IMDB

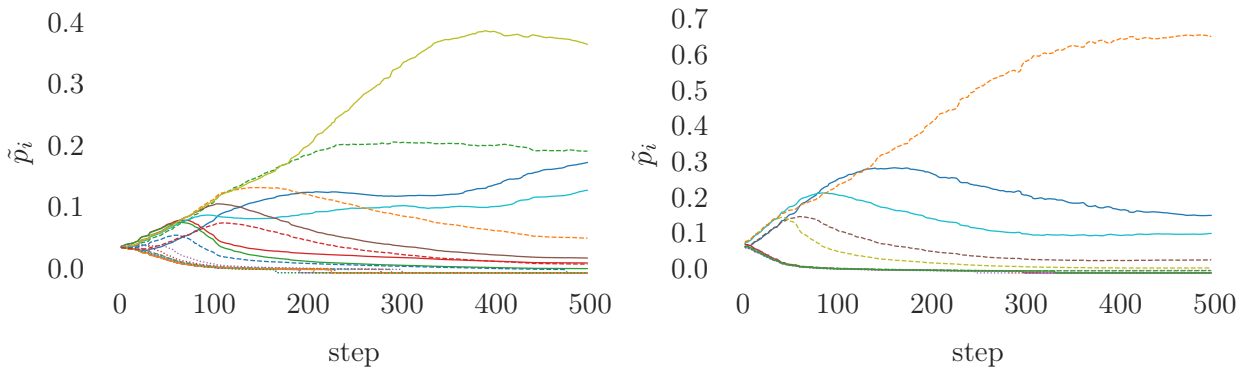
На наборе данных IMDB обучение проводилось с гиперпараметрами, приведёнными в таблице на рис. 7.

Параметр	**Значение (RoBERTa-large)**	**Значение (BART)**
init learning rate	10^{-4}	
weight decay	10^{-3}	
α	70	
λ	0	
distribution cutoff	$5 \cdot 10^{-3}$	
dropout	$3 \cdot 10^{-2}$	
attention heads	16	
k	64	
learning rate decay patience	5	
precision	amp fp16	
batch size	1200	588
add positional embeddings	no	
число параметров	1.59M	0.87M
доля параметров от базовой модели	**0.4%**	**0.2%**

Рис. 7: Гиперпараметры обучения на наборе данных IMDB и краткая информация о числе добавляемых параметров

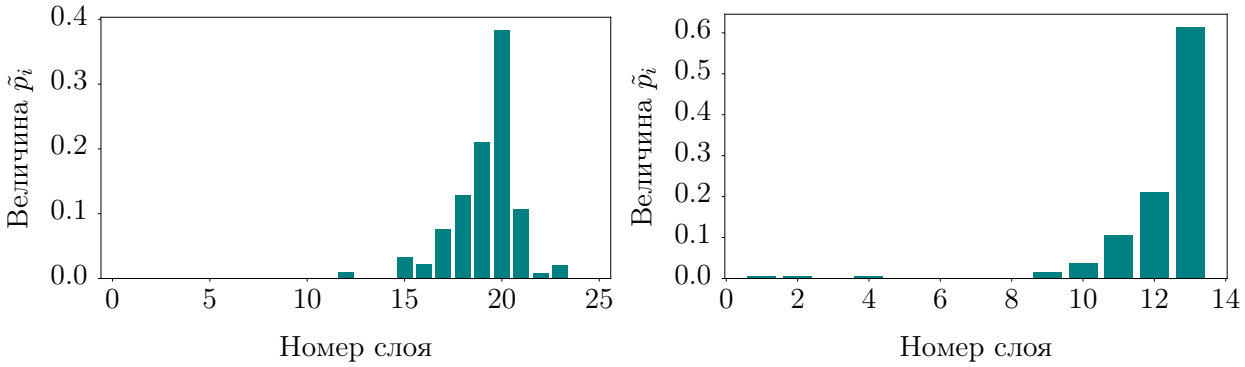
Модель выбиралась по максимальному качеству на валидации. На рис. 8 можно видеть, как менялись компоненты важности для каждого слоя в процессе обучения для обеих базовых моделей. Обученное распределение на слоях $\tilde{p}$ изображено на рис. 9.

Как можно заметить, модель выбрала некоторое количество слоёв среди последних. Низкоуровневые слои оказались занулены, что говорит об их малой значимости для задачи. Действительно, определение сентимента текста требует достаточно высокоуровневых признаков.



(а) RoBERTa-large (б) BART

Рис. 8: История изменения компонент вектора $\tilde{p}$ в процессе обучения у модели, обученной на задачу IMDB sentiment. Слева (8а) — базовая модель RoBERTa-large, справа (8б) — базовая модель BART.



(а) RoBERTa-large (б) BART

Рис. 9: Величины компонент вектора $\tilde{p}$ у модели, обученной на задачу IMDB sentiment. Модель с лучшими метриками на валидации. Слева (9а) — базовая модель RoBERTa-large, справа (9б) — базовая модель BART.

Для RoBERTa-large качество на тестовой части набора данных составило **96.1% по метрике accuracy**. Как видно из приложения 6.4, лучшие решения получают качество 96% - 96.68%. Таким образом, достигаемое качество предложенного алгоритма сопоставимо с лучшими решениями в области. Стоит заметить, что в этой работе базовая модель, в отличие от множества решений в приложении 6.4, не дообучалась. Более того, добавлено было всего лишь **0.4%** параметров от размера базовой модели.

Аналогично высокое качество получается и в случае базовой модели BART. Распределение влияния слоев представлено на рис. 9б. Достигаемое качество на тестовой части датасета — **96.0% по метрике accuracy**, что также располагает решение среди наилучших.

Некоторые примеры ошибочных предсказаний модели на основе RoBERTa-large представлены в приложении 6.5. При визуальном анализе многих примеров было замечено, что существенную часть ошибок составляют ошибки разметки или неоднозначные примеры — например, негативное содержание отзыва с позитивной оценкой или наоборот.

Для RoBERTa-large были обучены базовые решения (baseline). Для корректности сравнения была выбрана такая же архитектура, но используется принудительно лишь один из скрытых слоев: $p = \left(-\infty \quad \dots \quad \underset{i}{1} \quad \dots \quad -\infty\right)^T$. Так можно понять, насколько предложенное решение лучше простых вариантов, когда выбирается лишь один слой. Результаты представлены на рис. 10. Видно, что разрыв в качестве достаточно велик. Действительно, предложенный метод использует признаки с различных слоев, эффективно их комбинируя. Видно, что каждый отдельно взятый слой не дает того же качества, что и предлагаемый подход.

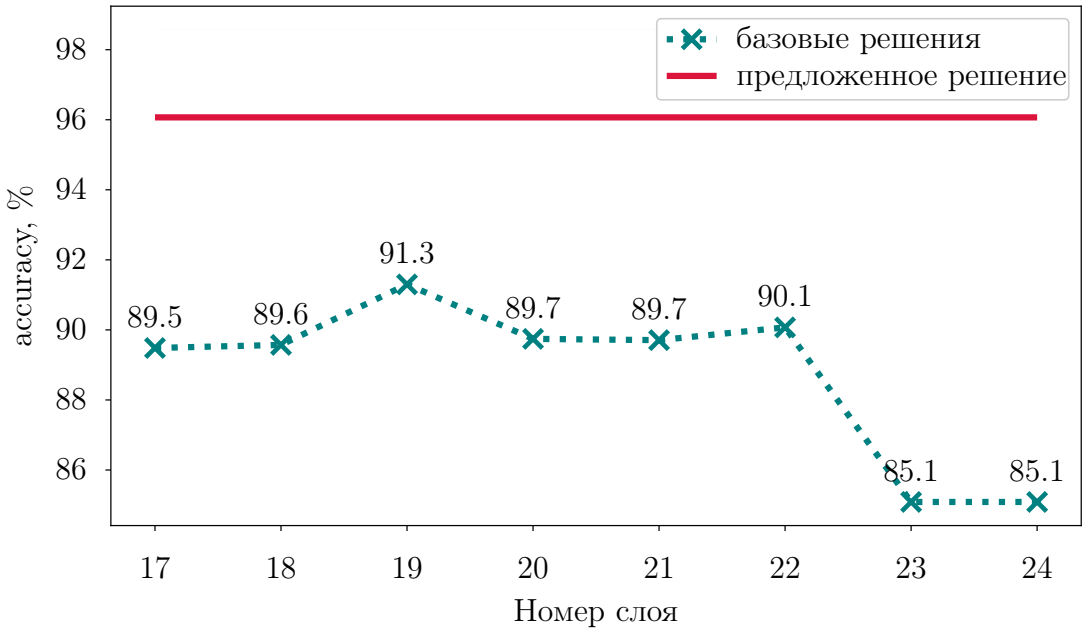


Рис. 10: Сравнение полученного решения с базовыми решениями — решениями, использующими состояния лишь одного скрытого слоя. Базовая модель: RoBERTa-large.

Таким образом, на датасете IMDB с обеими базовыми моделями получились результаты, сопоставимые с SOTA решениями. При этом важно заметить, что многие приведенные решения в приложении 6.4 обучали полностью новую модель. В предложенном решении базовые модели заморожены. Таким образом, можно сделать вывод, что алгоритм успешно справляется с данной задачей для обеих базовых моделей, при этом добавляя меньше одного процента параметров от размера базовой модели.

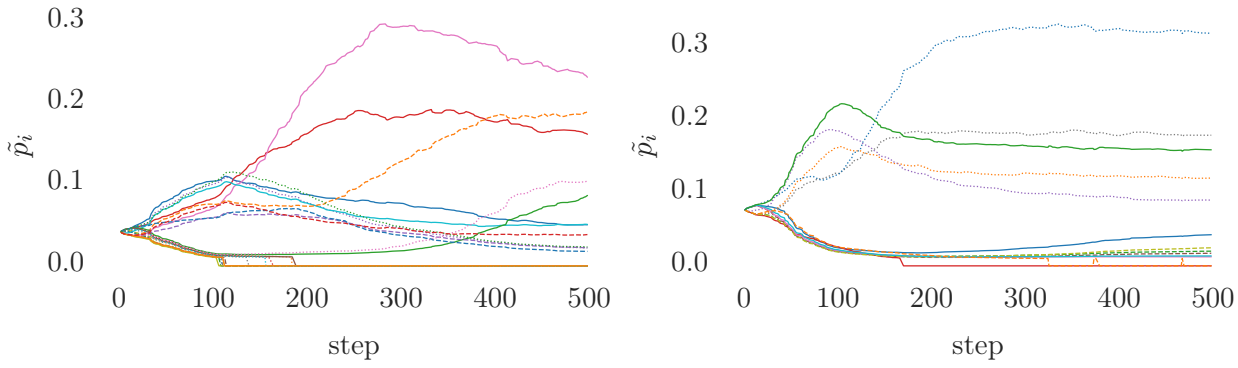
5.4.3 CoLA

Аналогичным образом, модель была обучена на датасете CoLA. На данном датасете проявился один из недостатков алгоритма — в условиях малого числа обучающих данных, модель достаточно просто переобучается. Поэтому необходимо включать дополнительные регуляризации и понижать число параметров. Гиперпараметры обучения представлены на рис. 11.

Модель выбиралась по максимальному качеству на валидации. Выученное распределение «важности» слоев изображено на рис. 13. В отличие от датасета IMDB, в полученном распределении выделились признаки с разных слоев.

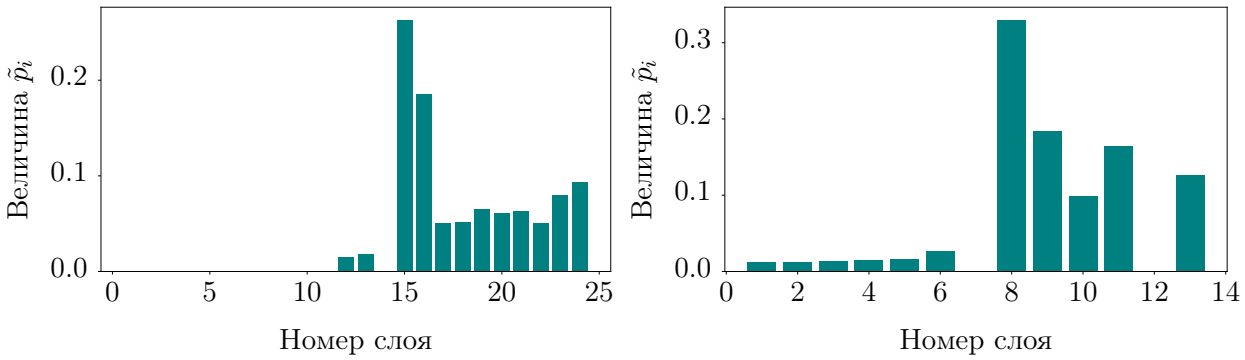
Параметр	**Значение (RoBERTa-large)**	**Значение (BART)**
init learning rate	10^{-4}	
weight decay	0	
α	64	
λ	0	
distribution cutoff	10^{-2}	
dropout	$3 \cdot 10^{-2}$	10^{-2}
attention heads	8	
k	64	
learning rate decay patience	25	16
precision	amp fp16	
batch size	600	900
add positional embeddings	yes	
число параметров	1.59M	0.87M
доля параметров от базовой модели	**0.4%**	**0.2%**

Рис. 11: Гипер-параметры обучения на наборе данных CoLA



(а) RoBERTa-large (б) BART

Рис. 12: История изменения компонент вектора $\tilde{p}$ в процессе обучения у модели, обученной на задачу CoLA. Слева (12а) — базовая модель RoBERTa-large, справа (12б) — базовая модель BART.



(а) RoBERTa-large (б) BART

Рис. 13: Величины компонент вектора $\tilde{p}$ у модели, обученной на задачу CoLA. Модель с лучшими метриками на валидации. Слева (13а) — базовая модель RoBERTa-large, справа (13б) — базовая модель BART.

Полученное качество также сопоставимо с лучшими решениями: **83.6% и 80.9% accuracy** для RoBERTa-large и BART соответственно.

На данном наборе данных предложенное решение также показывает отличные результаты. В отличие от IMDB, на CoLA видно различие в качестве для двух базовых моделей. RoBERTa-large, обученная на более общую задачу, получает качество выше. Такое различие можно объяснить тем, что при обучении на задачу суммаризации модель может начать игнорировать факторы, отвечающие за лингвистическую корректность текста, так как они не сильно важны для улавливания смысла.

Также интересным результатом является распределение на рис. 13. Видно, что сразу целый набор слоев имеет высокое влияние на результат. Это можно объяснить тем, что для определения ошибок в тексте необходимы как высокоуровневые признаки, так и факторы по специфике ближе к уровню слов.

На рис. 14 приведены базовые решения, использующие лишь один слой RoBERTa-large. Как видно, ни один из выбранных слоев по отдельности не дает качество такое же высокое, как предложенный алгоритм.

Сравнивая решение с приложением 6.4, видно, что лучшие решения имеют качество 82% - 88.6%. Таким образом, получаемое в работе качество находится в диапазоне лучших решений. Важно отметить, что среди приведенных решений нет подходов, не дообучающих базовую модель. Выходит, что предложенное решение без дообучения базовой модели успешно конкурирует с решениями, обучающими всю модель.

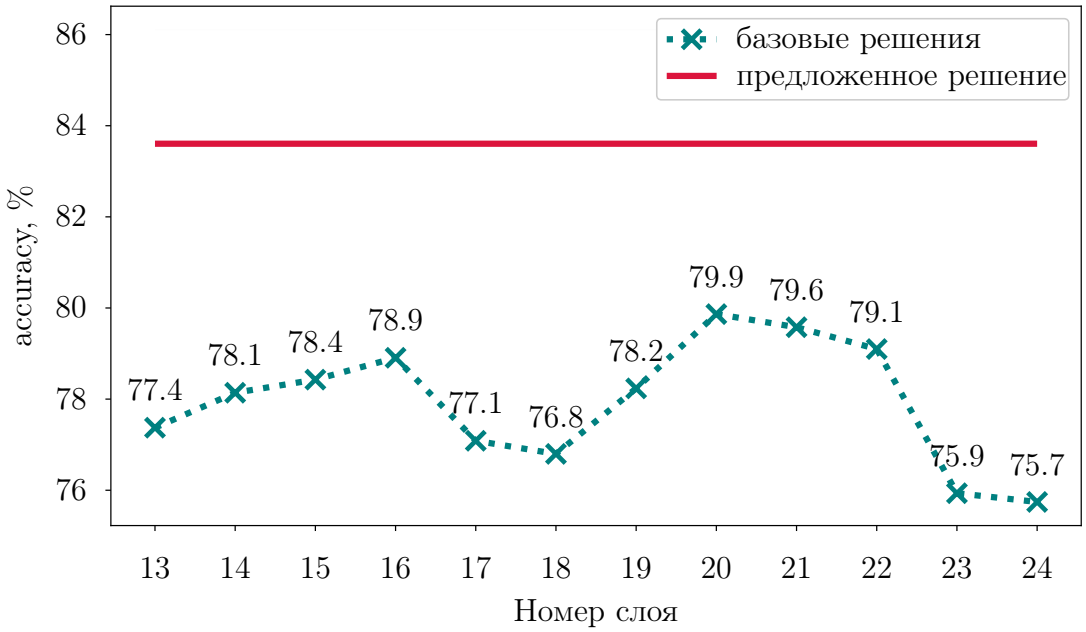


Рис. 14: Сравнение полученного решения с базовыми решениями — решениями, использующими состояния лишь одного скрытого слоя. Базовая модель: RoBERTa-large.

5.4.4 CoNLL

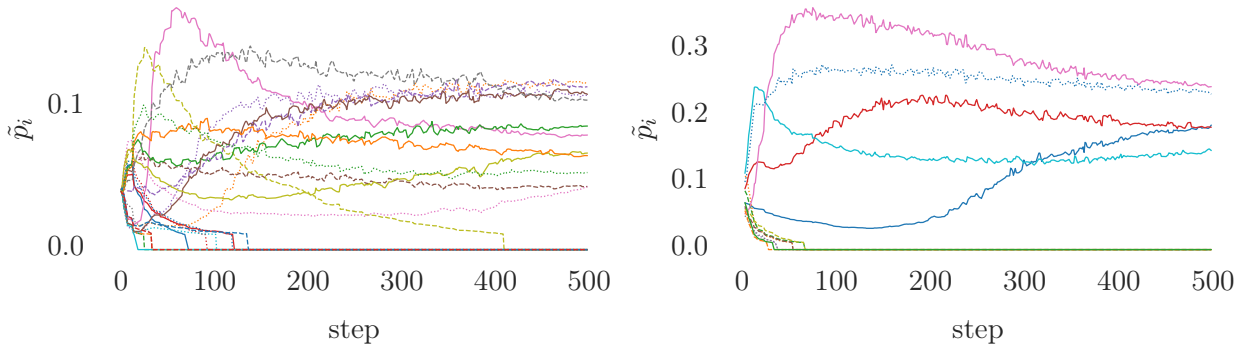
Последняя разбираемая задача, CoNLL, является задачей распознавания именных сущностей. В терминах нейросети эта задача эквивалентна классификации каждого входного токена. В отличие от предыдущих задач, для данного набора данных были использованы энкодерные слои в качестве финальных слоев над вектором признаков F.

Такая модификация обусловлена тем, что, в отличие от классификации всей последовательности, для классификации токенов требуется выучить дополнительные связи между токенами. Решения с одним слоем внимания на выходе, как в предыдущих пунктах, показывали не лучшие результаты. Гиперпараметры модели указаны в таблице на рис. 15.

Модель выбиралась по максимальному качеству на валидации. Выученное распределение «важности» слоев изображено на рис. 17. В приложенных распределениях видно, что модель использует как низкоуровневые признаки, так и более высокоуровневые. Это имеет смысл в контексте распознавания именованных сущностей. Для предсказания важны как признаки на уровне значения токенов, так и более высокоуровневые признаки.

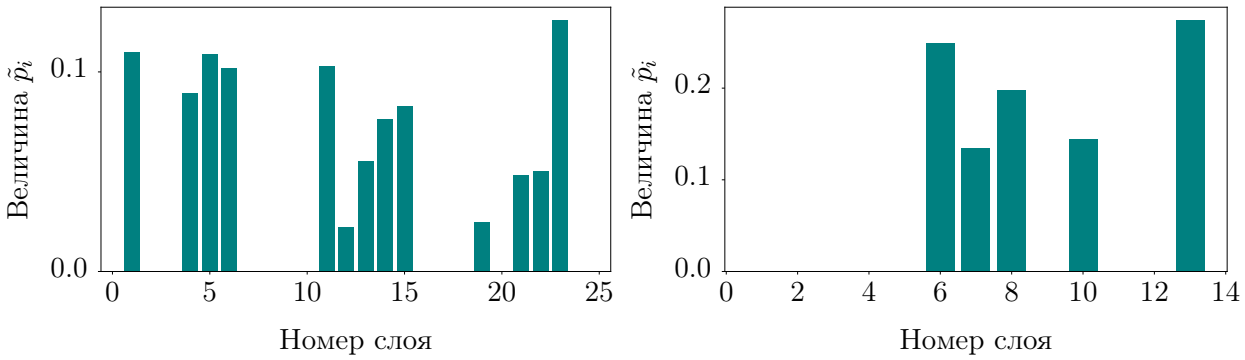
Параметр	**Значение (RoBERTa-large)**	**Значение (BART)**
init learning rate	10^{-4}	
weight decay	10^{-3}	
α	128	
λ	10^{-3}	
distribution cutoff	10^{-2}	
dropout	0.2	
attention heads	8	
k	128	
learning rate decay patience	16	
precision	amp fp16	
batch size	356	
encoder layers added	1	
dim feedforward	2048	
add positional embeddings	yes	
число параметров	3.21M	1.77M
доля параметров от базовой модели	**0.9%**	**0.4%**

Рис. 15: Гипер-параметры обучения на наборе данных CoLA



(а) RoBERTa-large (б) BART

Рис. 16: История изменения компонент вектора $\tilde{p}$ в процессе обучения у модели, обученной на задачу CoNLL. Слева (16а) — базовая модель RoBERTa-large, справа (16б) — базовая модель BART.



(а) RoBERTa-large (б) BART

Рис. 17: Величины компонент вектора $\tilde{p}$ у модели, обученной на задачу CoNLL. Модель с лучшими метриками на валидации. Слева (17а) — базовая модель RoBERTa-large, справа (17б) — базовая модель BART.

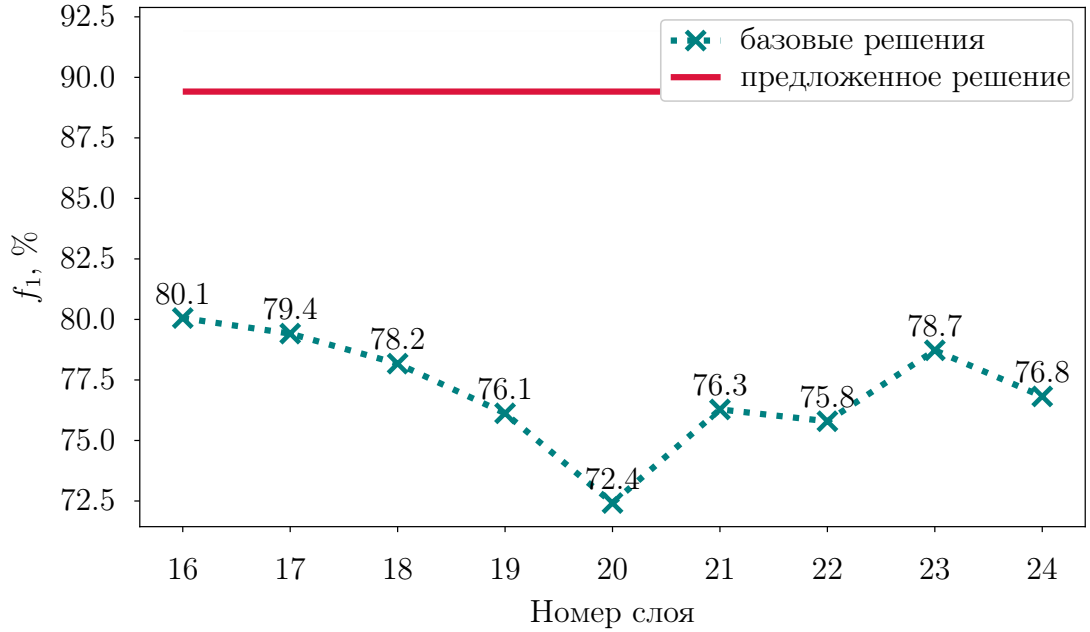


Рис. 18: Сравнение полученного решения с базовыми решениями — решениями, использующими состояния лишь одного скрытого слоя. Базовая модель: RoBERTa-large.

Полученное качество достаточно высоко: **89.4**% **и 89.3**% $\mathbf{f_1}$ для RoBERTa-large и BART соответственно. Сравнивая с приложением 6.4, где лучшие решения имеют f_1 93% - 94.6%, можно сделать вывод, что полученные результаты всё же ниже, чем лучшие решения. Однако по рис. 18 видно, что базовые решения, использующие лишь один слой RoBERTa-large по отдельности, получают качество значительно хуже качества предложенного решения.

В таблице на рис. 19 представлено качество NER для алгоритма с базовой моделью RoBERTa-large. Как видно, качество сильно просаживает один класс — MISC. Это можно объяснить тем, что модель не обучена на специфичные именованные сущности,

которые входят именно в этот класс.

Метка	**Precision**	**Recall**	$\mathbf{f_1}$	**Количество**
LOC	90.93%	92.51%	91.71	1697
MISC	72.44%	81.62%	76.76	791
ORG	85.05%	90.43%	87.66	1766
PER	94.02%	95.36%	94.69	1640

Рис. 19: Качество модели RoBERTa-large с предложенным алгоритмом на CoNLL: разбивка по классам именованных сущностей.

5.4.5 Общие наблюдения

Обучаемая модель оказалась достаточно робастна к изменениям гиперпараметров. Так, самыми влиятельными гиперпараметрами являются: k — размерность пространства скрытых слоев после проецирования, α — «скорость» изменения p.

Важно заметить, что при большом k алгоритм начинает быстро переобучаться. Поэтому в данной работе k используется относительно небольшой. Параметр α сильно влияет на скорость сходимости: при небольших значениях вклад малополезных слоев исчезает медленно, что может негативно влиять на качество. С другой стороны, высокие значения α провоцируют выделение важных слоев в самом начале тренировки, что может быть не оптимально.

6 Заключение

6.1 Результаты

В работе предложен алгоритм переиспользования вычислений основной трансформерной модели для решения смежных задач над теми же входными данными. Алгоритм протестирован на двух различных базовых моделях и на трех датасетах.

На всех тестах подход показывает высокие результаты, при этом на двух датасетах результат сопоставим с лучшими решениями. Важно заметить, что в предложенном алгоритме базовая модель не дообучается, и тем не менее предложенный подход успешно конкурирует с решениями, использующими полное дообучение модели.

Более того, предложенный подход эффективен в применении, как описано в секции 5.3. В частности, алгоритм можно применять эффективно по дополнительной памяти, что отличает его от схожих алгоритмов [21, 19].

Важнейшая деталь подхода в том, что базовая трансформерная модель остается неизменной. Таким образом, без негативного влияния на качество основной задачи, разработан эффективный алгоритм, способный решать смежные задачи с высоким качеством в условиях ограниченных вычислительных мощностей. Все цели работы достигнуты.

6.2 Исходный код и воспроизведение результатов

Исходный код выложен в открытый доступ в репозитории по ссылке: `https://github.com/alexdremov/adalayers`. В репозитории содержатся все конфигурационные файлы обучения, которые могут быть использованы для воспроизведения приведенных в работе результатов.

6.3 Вопросы для дальнейшего исследования

В работе были использованы две различные модели для проверки того, насколько качество алгоритма зависит от выбора базовой модели. Однако, в полной мере этот вопрос не был исследован. Так, например, видно, что модель RoBERTa-large, обученная на MLM, лучше справляется с NER, чем BART, обученная на суммаризацию. Также неизвестно, насколько сильно могут отличаться домены базовой задачи и смежной, чтобы появились весомые просадки в качестве.

Помимо первого вопроса, не до конца изучено влияние гиперпараметра α и вклада энтропийной регуляризации на параметр p. В ходе экспериментов и в данной работе было замечено, что эти параметры сильно влияют на качество, но отдельных экспериментов на выявление величины влияния не проводилось. Так как алгоритм склонен к переобучению при малом числе данных, более глубокая оценка влияния данных параметров важна.

Интересным направлением работы является применение алгоритма над признаками декодерных слоев, что допустимо в рамках предложенной архитектуры.

Список литературы

[1] *Vaswani, Ashish.* Attention Is All You Need. — 2023.

[2] *Devlin, Jacob.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. — 2019.

[3] Language Models are Unsupervised Multitask Learners / Alec Radford, Jeff Wu, Rewon Child et al. — 2019.

[4] *Raffel, Colin.* Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. — 2023.

[5] *Jacob, Benoit.* Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. — 2017.

[6] *Sanh, Victor.* DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. — 2020.

[7] *He, Tianxing.* Analyzing the Forgetting Problem in the Pretrain-Finetuning of Dialogue Response Models. — 2021.

[8] *Parović, Marinela.* Cross-Lingual Transfer with Target Language-Ready Task Adapters. — 2023.

[9] *Lialin, Vladislav.* Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning. — 2023.

[10] *Houlsby, Neil.* Parameter-Efficient Transfer Learning for NLP. — 2019.

[11] *Wang, Yaqing.* AdaMix: Mixture-of-Adaptations for Parameter-efficient Model Tuning. — 2022.

[12] *Zaken, Elad Ben.* BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. — 2022.

[13] *Guo, Demi.* Parameter-Efficient Transfer Learning with Diff Pruning. — 2021.

[14] *Vucetic, Danilo.* Efficient Fine-Tuning of BERT Models on the Edge. — 2022. `http://dx.doi.org/10.1109/ISCAS48785.2022.9937567`.

[15] *Aghajanyan, Armen.* Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. — 2020.

[16] *Hu, Edward J.* LoRA: Low-Rank Adaptation of Large Language Models. — 2021.

[17] *Lester, Brian.* The Power of Scale for Parameter-Efficient Prompt Tuning. — 2021.

[18] *Xie, Shuo.* Hidden State Variability of Pretrained Language Models Can Guide Computation Reduction for Transfer Learning. — 2022.

[19] *Heinsen, Franz A.* An Algorithm for Routing Vectors in Sequences. — 2022.

[20] *Sabour, Sara.* Dynamic Routing Between Capsules. — 2017.

[21] *Evci, Utku.* Head2Toe: Utilizing Intermediate Representations for Better Transfer Learning. — 2022.

[22] RoBERTa: A Robustly Optimized BERT Pretraining Approach / Yinhan Liu, Myle Ott, Naman Goyal et al. // *CoRR.* — 2019. — Vol. abs/1907.11692. `http://arxiv.org/abs/1907.11692`.

[23] BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension / Mike Lewis, Yinhan Liu, Naman Goyal et al. // *CoRR.* — 2019. — Vol. abs/1910.13461. `http://arxiv.org/abs/1910.13461`.

[24] Learning Word Vectors for Sentiment Analysis / Andrew L. Maas, Raymond E. Daly, Peter T. Pham et al. // Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. — Portland, Oregon, USA: Association for Computational Linguistics, 2011. — June. — Pp. 142–150. `http://www.aclweb.org/anthology/P11-1015`.

[25] *Warstadt, Alex.* Neural Network Acceptability Judgments / Alex Warstadt, Amanpreet Singh, Samuel R Bowman // *arXiv preprint arXiv:1805.12471.* — 2018.

[26] *Rücker, Susanna.* CleanCoNLL: A Nearly Noise-Free Named Entity Recognition Dataset. — 2023.

[27] *Clark, Kevin.* What Does BERT Look At? An Analysis of BERT's Attention. — 2019.

[28] *Coenen, Andy.* Visualizing and Measuring the Geometry of BERT. — 2019.

[29] *Vig, Jesse.* Analyzing the Structure of Attention in a Transformer Language Model. — 2019.

[30] *Hendrycks, Dan.* Gaussian Error Linear Units (GELUs). — 2023.

[31] *Xiong, Ruibin.* On Layer Normalization in the Transformer Architecture. — 2020.

[32] *Paszke, Adam.* PyTorch: An Imperative Style, High-Performance Deep Learning Library. — 2019.

[33] *Kingma, Diederik P.* Adam: A Method for Stochastic Optimization. — 2017.

[34] Dropout: A Simple Way to Prevent Neural Networks from Overfitting / Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky et al. // *Journal of Machine Learning Research*. — 2014. — Vol. 15, no. 56. — Pp. 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.

[35] SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems / Alex Wang, Yada Pruksachatkun, Nikita Nangia et al. // *arXiv preprint 1905.00537*. — 2019.

[36] Acceptability Judgements via Examining the Topology of Attention Maps / Daniil Cherniavskii, Eduard Tulchinskii, Vladislav Mikhailov et al. // Findings of the Association for Computational Linguistics: EMNLP 2022. — Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022. — dec. — Pp. 88–107. https://aclanthology.org/2022.findings-emnlp.7.

[37] *Proskurina, Irina*. Can BERT eat RuCoLA? Topological Data Analysis to Explain / Irina Proskurina, Ekaterina Artemova, Irina Piontkovskaya // Proceedings of the 9th Workshop on Slavic Natural Language Processing 2023 (SlavicNLP 2023). — Association for Computational Linguistics, 2023. http://dx.doi.org/10.18653/v1/2023.bsnlp-1.15.

[38] *Sileo, Damien*. tasksource: A Dataset Harmonization Framework for Streamlined NLP Multi-Task Learning and Evaluation. — 2023.

[39] *Wang, Sinong*. Entailment as Few-Shot Learner. — 2021.

[40] *Charpentier, Lucas Georges Gabriel*. Not all layers are equally as important: Every Layer Counts BERT. — 2023.

[41] *Lee-Thorp, James*. FNet: Mixing Tokens with Fourier Transforms. — 2022.

[42] *Wang, Xinyu*. Automated Concatenation of Embeddings for Structured Prediction. — 2021.

[43] *Yamada, Ikuya*. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. — 2020.

[44] *Zhou, Wenxuan*. Learning from Noisy Labels for Entity-Centric Information Extraction / Wenxuan Zhou, Muhao Chen // Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. — Association for Computational Linguistics, 2021. http://dx.doi.org/10.18653/v1/2021.emnlp-main.437.

[45] *Liu, Tianyu*. Autoregressive Structured Prediction with Language Models. — 2022.

[46] *Schweter, Stefan*. FLERT: Document-Level Features for Named Entity Recognition. — 2021.

[47] *Ye, Deming.* Packed Levitated Marker for Entity and Relation Extraction. — 2022.

[48] *Wang, Xinyu.* Improving Named Entity Recognition by External Context Retrieving and Cooperative Learning. — 2022.

[49] Named Entity Recognition Architecture Combining Contextual and Global Features / Tran Thi Hong Hanh, Antoine Doucet, Nicolas Sidere et al. // Towards Open and Trustworthy Digital Societies. — Springer International Publishing, 2021. — P. 264–276. http://dx.doi.org/10.1007/978-3-030-91669-5_21.

[50] *Csanády, Bálint.* LlamBERT: Large-scale low-cost data annotation in NLP. — 2024.

[51] *Yang, Zhilin.* XLNet: Generalized Autoregressive Pretraining for Language Understanding. — 2020.

[52] *Haonan, Lu.* Graph Star Net for Generalized Multi-Task Learning. — 2019.

Приложение

6.4 Качество лучших моделей на выбранных наборах данных

- **IMDB Sentiment** [24], accuracy.

  1. **96.68%** — RoBERTa-large with LlamBERT [50]
  2. **96.54%** — RoBERTa-large [50]
  3. **96.21%** — XLNet [51]
  4. **96.2%** — Heinsen Routing + RoBERTa Large [19]
  5. **96.1%** — RoBERTa-large 355M + Entailment as Few-shot Learner [39]
  6. **96.0%** — GraphStar [52]

- **GLUE CoLA** [25], accuracy.

  1. **88.6%** — En-BERT + TDA + PCA [36]
  2. **88.2%** — BERT+TDA [37]
  3. **87.3%** — RoBERTa+TDA [37]
  4. **87.15%** — deberta-v3-base+tasksource [38]
  5. **86.4%** — RoBERTa-large 355M + Entailment as Few-shot Learner [39]
  6. **82.7%** — LTG-BERT-base 98M [40]
  7. **82.6%** — ELC-BERT-base 98M [40]
  8. **82.1%** — En-BERT + TDA [36]
  9. **78%** — FNet-Large [41]

- **CoNLL** [26] NER, f_1.

  1. **94.6%** — ACE + document-context [42]
  2. **94.3%** — LUKE 483M [43]
  3. **94.22%** — Co-regularized LUKE [44]
  4. **94.1%** — ASP+T5-3B [45]
  5. **94.09%** — FLERT XLM-R [46]
  6. **94.0%** — PL-Marker [47]
  7. **93.85%** — CL-KL [48]
  8. **93.82%** — XLNet-GCN [49]
  9. **93.8%** — ASP+flan-T5-large [45]

6.5 Примеры ошибок модели на датасете IMDB

Далее приведены некоторые примеры элементов из датасета IMDB, истинная метка и предсказание.

> First off let me say, If you haven't enjoyed a Van Damme movie since bloodsport, you probably will not like this movie. Most of these movies may not have the best plots or best actors **but I enjoy these kinds of movies** for what they are [...] **Good fun stuff!**

Истинная метка: negative
Предсказание: positive

> Just watched on UbuWeb this early experimental short film directed by William Vance and Orson Welles. Yes, you read that right, Orson Welles! [...] I won't reveal any more except to say **how interesting the silent images were** as they jump-cut constantly. That's not to say this was **any good but it was fascinating** to watch even with the guitar score (by Larry Morotta) [...]

Истинная метка: negative
Предсказание: positive

> The theme is controversial and the depiction of the hypocritical and sexually starved india is excellent.**Nothing more to this film**. There is a **lack of good dialogues**(why was the movie in english??). There was **lack of continuity** and **lack of passion/emotion** in the acting.

Истинная метка: positive
Предсказание: negative

> If you want to see a brilliant performance of Mikado, played to perfection with expert timing and panache, **don't watch this version.** [...] It's a lot of fun and a good intro to Gilbert and Sullivan, but after this, **rush out and rent the Canadian Stratford version**. You'll see the difference between **good and great**. Nobody does G&S better than Brian McDonald and the Stratford group.

Истинная метка: positive
Предсказание: negative