Федеральное государственное автономное образовательное учреждение
высшего образования «Московский физико-технический институт
(национальный исследовательский университет)»
Кафедра компьютерной лингвистики
Лаборатория нейронных систем и глубокого обучения

На правах рукописи

Куратов Юрий Михайлович

Специализация языковых моделей для применения к задачам обработки естественного языка

Специальность 05.13.17 —
«Теоретические основы информатики»

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
кандидат физико-математических наук
Бурцев Михаил Сергеевич

Москва — 2020

Оглавление

Стр.

Стр.

Стр.

Введение

Диссертационная работа посвящена методам обучения языковых моделей и в частности методам переноса знаний при дообучении языковых моделей на данных, ограниченных тематикой или заданным набором языков.

В области обработки естественного языка методы машинного и глубокого обучения стали широко применимы за последние десятилетия, но обычно требуют большого числа размеченных данных для получения высоких результатов. Доступные неразмеченные данные могут быть эффективно использованы для обучения векторных представлений слов на основе контекстно-независимых [1—4] и контекстно-зависимых языковых моделей [5—8]. Обучение языковых моделей требовательно к вычислительным ресурсам, при этом увеличение числа параметров только улучшает их качество работы [7—10] (`BERT-Base` — 110 миллионов параметров, 4 дня вычислений на 4 Cloud TPUv2[1], `BERT-Large` — 340 миллионов параметров, 4 дня на 16 Cloud TPUv2, `GPT-2` — 1,5 миллиарда параметров, 256 Cloud TPUv3, MegatronLM [11] — 8,3 миллиарда параметров, 512 NVIDIA V100 32GB GPU).

В публичном доступе обычно появляются модели для английского языка или модели обученные сразу на большом числе языков (`BERT`, `многоязычный BERT`). Многоязычные модели универсальны, но уступают по качеству моделям, обученным специально под один язык. Однако, обучать модели под каждый язык ресурсозатратно. Отсюда возникает **проблема** настоящего исследования: как обучать языко- и доменно- специфичные модели максимально переиспользуя уже обученные модели.

В ряде работ обучение языковых моделей для других языков повторяет процесс обучения моделей для английского языка и требует больших объемов данных и доступных вычислительных ресурсов, например, CamemBERT [12] и FlauBERT [13] для французского языка. Были предприняты попытки использования методов переноса знаний для обучения модели SciBERT [14] на коллекции научных публикаций, где в качестве инициализации использовались веса предобученной модели `BERT`. Также в этой работе была показана важность использования словаря адаптированного под тематику текстов, но обучение модели с измененным словарем производилось уже со случайной инициализации

[1] https://cloud.google.com/tpu

параметров, т. е. без переноса знаний. Метод переноса знаний применяется для обучения языковых моделей для двух языков — язык на котором уже обучена модель и новый, при этом обучаются только векторные представления слов [15]. Разрабатываются и методы обучения многоязычных моделей, почти не уступающих в качестве одноязычным [16; 17], но обучение таких моделей также происходит без переиспользования уже существующих моделей.

В связи с этим была обозначена проблема исследования и сформулирована цель диссертационной работы.

Целью данной работы является исследование переноса знаний при дообучении языковых моделей на данных, ограниченных тематикой или заданным набором языков, а также оценка эффективности полученных моделей при решении различных задач обработки естественного языка.

Для достижения поставленной цели были определены и решены следующие основные **задачи**.

1. Предложить подход и разработать метод дообучения для переноса знаний с предобученных языковых моделей на другой домен или другой набор языков.
2. Применить разработанный метод переноса знаний для обучения языковых моделей для русского языка и группы славянских языков — болгарского, чешского, польского и русского.
3. Применить разработанный метода переноса знаний для обучения языковых моделей разговорного домена для русского и английского языков.
4. Разработать и опубликовать в открытом доступе программный комплекс для дообучения языковых моделей.
5. Исследовать эффективность полученных моделей на задачах классификации, разметки последовательности, разрешения кореференции и поиска ответа на вопрос в тексте.

Научная новизна:

1. Был предложен оригинальный метод переноса знаний с предобученных языковых моделей с использованием пересборки словаря под языковую модель, на которую осуществляется перенос знаний.
2. Впервые было проведено исследование эффективности языковых моделей на базе архитектуры Трансформер, предобученных специально для русского языка, для решения задач классификации текстов, раз-

метки последовательности, разрешения кореференции и поиска ответа на вопрос в тексте.

Теоретическая и практическая значимость. Теоретическая значимость диссертационной работы заключается в следующих положениях:

1. Предложен метод дообучения для переноса знаний с предобученных языковых моделей на языковые модели других доменов или языков;
2. Показано, что дообучение языковых моделей под выбранный язык или домен позволяет получать более высокие оценки качества на целевых задачах, по сравнению с исходными моделями.

К практической значимости относятся следующие положения.

1. Предложенный метод переноса знаний позволяет ускорить процесс дообучения языковых моделей.
2. С помощью предложенного метода переноса знаний обучены: языковая модель для русского языка, языковая модель для болгарского, чешского, польского и русского языков, языковая модель для русского языка разговорного домена и языковая модель для английского языка разговорного домена.
3. Для задач классификации, разметки последовательности, разрешения кореференции и поиска ответа на вопрос в тексте установлены новые максимальные значения показателей качества, определяющие текущий статус научного прогресса.
4. Обученные в рамках диссертационной работы языковые модели выложены в открытый доступ и могут быть использованы для улучшения решений для русского, болгарского, чешского, польского языков, а также для разговорного домена для английского и русского языков.
5. Обученные в рамках диссертационной работы модели для решения задач классификации, разметки последовательности и поиска ответа на вопрос в тексте выложены в открытый доступ и готовы для использования в приложениях.

Методология и методы исследования. В ходе работы была применена методология численного эксперимента для исследования рассматриваемых в диссертации задач, применены методы теории вероятностей, машинного обучения и теории нейронных сетей. Также были применены методы разработки приложений на языке программирования Python, языке для написания скриптов Bash, программной библиотеке для машинного обучения TensorFlow.

Основные положения, выносимые на защиту:

1. Предложенный метод дообучения, основанный на пересборке словаря и матрицы векторных представлений, позволяет ускорить процесс дообучения языковых моделей.
2. Дообучение языковых моделей под заданный домен или язык позволяет улучшить качество решения задач на этом домене или языке.

Достоверность полученных результатов обеспечивается методикой проведения численных экспериментов. Код и параметры обученных моделей выложены в открытый доступ в составе библиотеки DeepPavlov и в репозиториях организации deepmipt[2]. Это позволяет воспроизвести результаты экспериментов, проведенных в данной работе. Результаты диссертации находятся в соответствии с результатами, полученными другими авторами для других языков, доменов и задач.

Апробация работы. Основные результаты работы данной диссертации докладывались на международных конференциях и семинарах:

- The 7th Workshop on Balto-Slavic Natural Language Processing in conjunction with 57th Annual Meeting of the ACL: Association for Computational Linguistics, 2nd August 2019, Florence, Italy;
- XXV Международная конференция по компьютерной лингвистике и интеллектуальным технологиям «Диалог», 29 мая – 1 июня 2019, Москва;
- The 56th Annual Meeting of the Association for Computational Linguistics, Systems Demonstrations, 15 – 20 July 2018, Melbourne, Australia;
- XXV Международная научная конференция студентов, аспирантов и молодых ученых «Ломоносов», 9 – 13 апреля 2018, Москва.

Кроме того, модели, полученные в результате работы над диссертацией, добавлены в библиотеку DeepPavlov и находят свое применение у ее пользователей (например, `RuBERT` был скачан 5648 раз с 4 июля по 3 августа 2020 г. через каталог моделей HuggingFace[3]).

Публикации. Основные результаты по теме диссертации изложены в 5 печатных изданиях, 3 из которых изданы в периодических научных журналах, индексируемых Web of Science и Scopus, 2 — в тезисах докладов.

Личный вклад. Результаты, доложенные на конференции [18], полностью получены автором диссертации. В работе [19] автором было произведено

[2] https://github.com/deepmipt

[3] https://huggingface.co/DeepPavlov/rubert-base-cased

обучение языковых моделей предложенным методом переноса знаний и проведены все эксперименты, Архипов Михаил реализовал первую версию программного кода для пересборки словаря, публично доступная версия пересборки словаря реализована автором. В работе [20] автором, совместно с Петровым Максимом, была реализована базовая модель, адаптирована базовая модель для русского языка, проведены эксперименты с языковыми моделями, собраны итоговые решения для соревнования по разрешению кореференции и анафоры «Dialogue Evaluation 2019», Ле Тхе Ань проводил эксперименты с моделью, использующей кореферентные связи между предложений. В работе [21] автором реализованы модели для поиска ответа на вопрос в тексте. В работе [22] автором была обучена предложенным методом переноса знаний модель `Славянский BERT`, которая позволила занять первое место по двум из трех метрик на соревновании «BSNLP 2019 Shared Task», Архиповым Михаилом и Трофимовой Марией были реализованы модели для распознавания именованных сущностей и проведены эксперименты с использованием модели `Славянский BERT`, Сорокин Алексей реализовал алгоритмы для нормализации извлеченных сущностей.

Объем и структура работы. Диссертация состоит из введения, пяти глав, заключения и двух приложений. Полный объём диссертации составляет 121 страницу, включая 31 рисунок и 19 таблиц. Список литературы содержит 135 наименований.

Глава 1. Языковые модели в задачах обработки естественного языка

1.1 Языковые модели

Языковая модель (language model) моделирует вероятностное распределение последовательности слов $P(w_1, w_2, ..., w_N)$, т. е. позволяет сказать с какой вероятностью может встретиться последовательность слов $w_1, w_2, ..., w_N$. Также языковым моделированием (language modeling) называют задачу предсказания следующего слова в последовательности $P(w_N|w_1, w_2, ..., w_{N-1})$ (рисунок 1.1a). Поэтому в литературе языковой моделью называют модели описывающие как распределение $P(w_1, w_2, ..., w_N)$, так и распределение $P(w_N|w_1, w_2, ..., w_{N-1})$ [23, с. 83, 957] [24, с. 219]. Оба определения языковой модели связаны друг с другом, так как по определению условной вероятности можно расписать:

$$P(w_1, w_2, \ldots, w_N) = P(w_1)P(w_2|w_1) \ldots P(w_N|w_1,w_2, \ldots ,w_{N-1}). \tag{1.1}$$

На рисунке 1.3 изображена языковая модель на основе рекуррентной нейронной сети, которая слева направо обрабатывает входную последовательность $w_1, w_2, ..., w_{N-1}$ и предсказывает распределение $P(w_N|w_1, w_2, ..., w_{N-1})$. Существуют и более простые языковые модели, основанные на N-граммах, т. е. учитывающие только N слов (-грамм, токенов). Например, для языковой модели на униграммах выражение из формулы 1.1 можно далее расписать как:

$$P(w_1, w_2, \ldots, w_N) = P(w_1) \cdot P(w_2) \cdot \ldots \cdot P(w_N).$$

Соответственно, для биграммной языковой модели:

$$P(w_1, w_2, \ldots, w_N) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_2) \cdot \ldots \cdot P(w_N|w_{N-1}),$$

т. е. соблюдается Марковское свойство первого порядка для униграмм и второго порядка для биграмм.

В 2018 году была предложена задача маскированного языкового моделирования (masked language model, MLM) [8], которая стала широко распространена для предобучения языковых моделей (подробнее про предобучение языковых моделей в разделах 1.2, 2.2). Задача маскированного языкового моделирования

состоит в предсказании распределения для слова в позиции i по всем словам в левом и правом контексте:

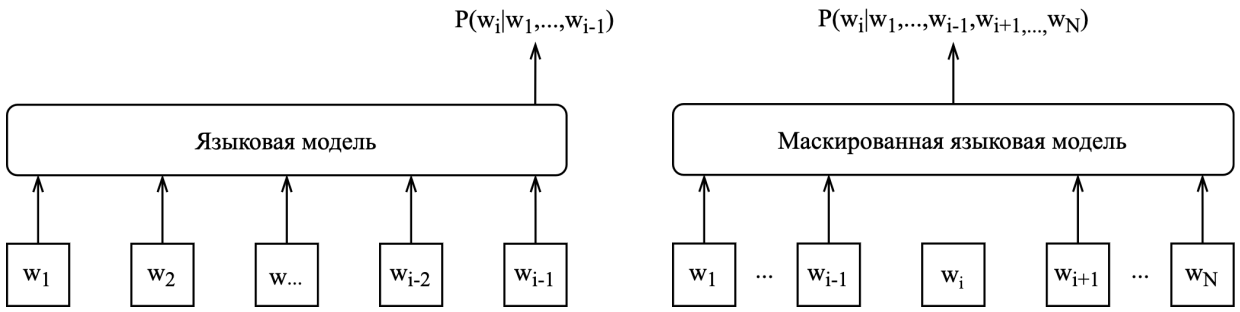
$$P(w_i|w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_N),$$

где весь текст состоит из N слов. Модель изображена на рисунке 1.1б.

Для того, чтобы вести рассуждения и о задаче маскированного языкового моделирования, и о задаче языкового моделирования (предсказания следующего слова) в данной работе под языковой моделью будет пониматься более широкий класс моделей:

$$P(w_i|w_{i-c_L}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+c_R}), \tag{1.2}$$

где c_L и c_R — длины левого и правого контекста для слова на позиции i. Принятое в литературе определение языковой модели $P(w_N|w_1, w_2, ..., w_{N-1})$ (и формула 1.1) будет ее частным случаем с $c_R = 0$, $c_L = N - 1$, $i = N$, а для маскированного языкового моделирования (MLM) — $c_R = i - 1$, $c_L = N - i$. На рисунке 1.1 изображены языковые модели предсказывающие следующее слово и маскированные языковые модели.



а) Языковая модель, предсказывающая следующее слово.

б) Маскированная языковая модель.

Рисунок 1.1 — Языковые модели.

Первые применения нейронных сетей для языкового моделирования работали на уровне униграмм, т. е. предсказывали следующее слово только по текущему [25]. Затем, были применены нейронные сети прямого распространения (Feed-Forward Neural Networks), учитывающие контекст фиксированной длины [26]. Вектора слов конкатенировались и подавались на вход полносвязному слою, а затем формировалось предсказание с помощью слоя с softmax функцией активации (рисунок 1.2).

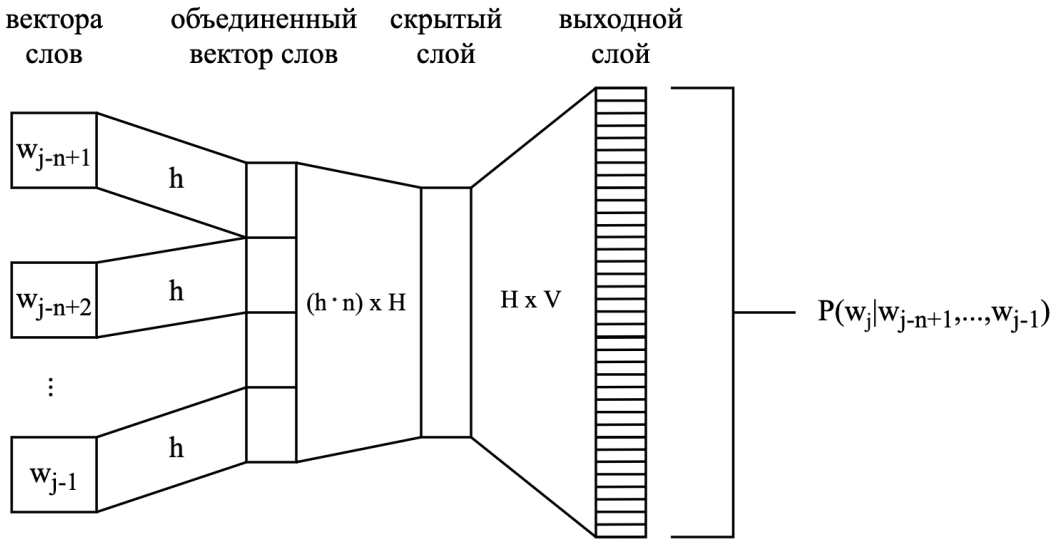


Рисунок 1.2 — Нейронная сеть прямого распространения (FFNN) для задачи языкового моделирования [26]. h — размерность векторного представления слова, n — длина контекста, H — размерность скрытого слоя, V — размер словаря.

Основным недостатком предыдущей модели была фиксированная длина контекста. Рекуррентные нейронные сети (recurrent neural network, RNN) могут работать с последовательностями разных длин, что позволяет избавиться от фиксированной длины контекста и применять их для языкового моделирования (рисунок 1.3) [27].

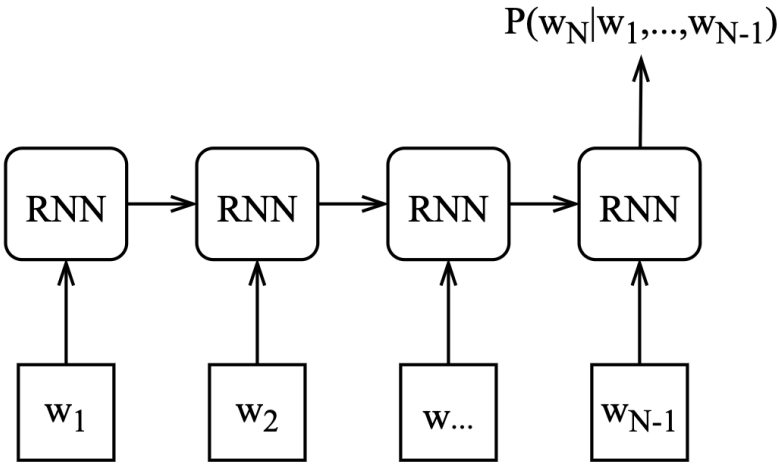


Рисунок 1.3 — Языковая модель на основе рекуррентной нейронной сети.

В рекуррентных сетях могут быть использованы LSTM [28] и GRU [29] ячейки для языкового моделирования [30]. Позже были предложены различные оптимизированные варианты. Например, QRNN [31] — вариант сверточной сети для языкового моделирования, которая работает быстрее и не уступает LSTM по качеству. AWD-LSTM [32] — вариант LSTM с добавлением регуляризации на скрытые состояния (DropConnect) для решения проблемы переобучения.

Появившаяся позже архитектура Трансформер [33] также применяется для языкового моделирования [7].

Общая архитектура нейросетевой языковой модели (рисунок 1.4) не изменяется при использовании рекуррентных, сверточных или сетей на основе Трансформер.

Каждое слово во входной последовательности $w_1, w_2, ..., w_{N-1}$ представляется в виде векторов $e_1, e_2, ..., e_{N-1}$ с помощью проекционного слоя (projection layer), который представляет из себя матрицу E векторных представлений слов. Каждая строка i такой матрицы является векторным представлением e_{w_i} для слова w_i. Размер матрицы $|V| \times d$, где V — словарь, d — размерность векторного представления. Для простоты обозначений будем писать e_i вместо e_{w_i}. Далее, все вектора $e_1, e_2, ..., e_{N-1}$ подаются в кодировщик (encoder), который возвращает вектор h_{N-1} размера h. Для рекуррентной языковой модели вектор h_{N-1} может быть скрытым состоянием сети на шаге $N-1$, для языковой модели на основе Трансформер — выход с последнего Трансформер слоя для позиции $N-1$. Затем, вектор h_{N-1} используется в проекционном слое:

$$logits = proj(FC(h_{N-1})) = FC(h_{N-1}) \cdot E^\top + b,$$

где b — вектор смещения (bias). Перед использованием вектора h_{N-1} в проекционном слое (projection layer) к нему может быть применен полносвязный слой FC, если размерность h неравна d или для увеличения глубины модели. Параметры проекционного слоя обычно общие с первым проекционным слоем и являются параметрами матрицы E. Это делается для уменьшения общего числа параметров модели, так как размерности d порядка 100 – 1000, а размеры словаря V имеют порядок 10^4 – 10^6. В итоге, распределение вероятностей следующего слова вычисляется как:

$$P(w_N|w_1, w_2, ..., w_{N-1}) = softmax(logits) = softmax(FC(h_{N-1}) \cdot E^\top + b).$$

1.1.1 Sequence-to-sequence модели

Если взять языковую модель, предсказывающую следующее слово, то можно с помощью нее генерировать продолжение текста. Для последовательности слов $w_1, w_2, \ldots, w_i$ языковая модель предсказывает распределение

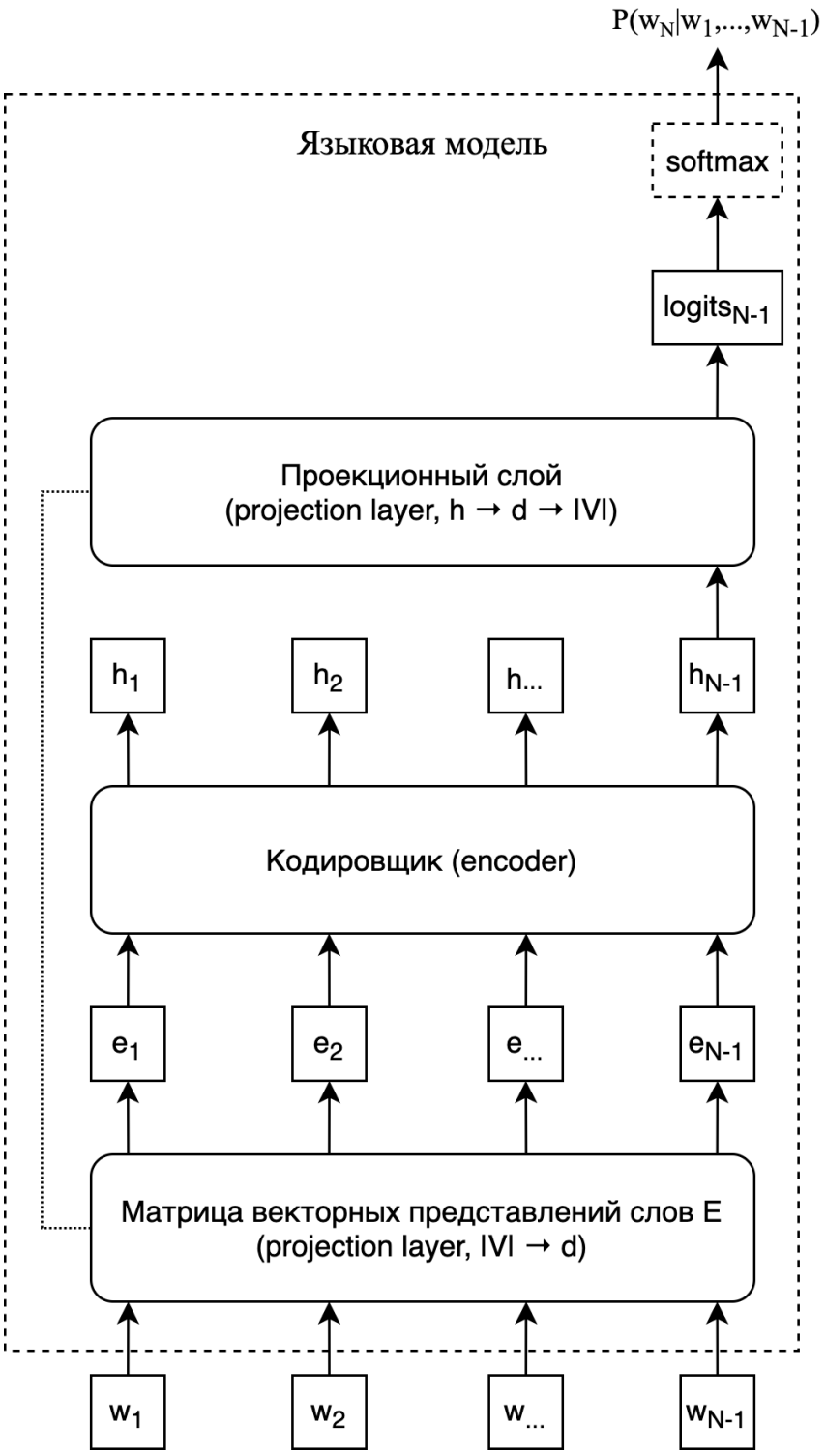


Рисунок 1.4 — Схема нейросетевой языковой модели. $w_1, w_2, ..., w_{N-1}$ – входная последовательность слов. E – матрица векторных представлений слов из которой формируются вектора $e_1, e_2, ..., e_{N-1}$ (первый проекционный слой). Затем вектор h_{N-1} из выхода кодировщика подается во второй проекционный слой для определения распределения следующего слова в словаре. $|V|$ – размер словаря, d – размерность векторных представлений слов, h – размерность выхода кодировщика.В обоих проекционных слоях зачастую используется одна и та же матрица векторных представлений слов E для уменьшения числа параметров языковой модели.

$P(w_{i+1}|w_1, w_2, ..., w_i)$. Из этого распределения можно взять, например, жадным образом наиболее вероятное предсказание и добавить к начальной последовательности, образуя новую последовательность $w_1, w_2, \ldots, w_i, w_{i+1}$. Таким образом можно продолжить процесс генерации текста [34].

Если в качестве начальной последовательности слов взять предложение на одном языке и обучать модель генерировать ее перевод на другой язык, то получится sequence-to-sequence (последовательность в последовательность, кодировщик-декодировщик) [35] модель. Sequence-to-sequence модель состоит из кодировщика (encoder, энкодер) и декодировщика (decoder, декодер) и изображена на рисунке 1.5. Sequence-to-sequence модели были изначально применены для задачи машинного перевода: кодировщик обрабатывает текст на исходном языке (source), а декодировщик, используя информацию от кодировщика, генерирует перевод на целевой язык (target). Кодировщик и декодировщик могут быть реализованы с помощью рекуррентных [35], сверточных сетей [36] или архитектуры Трансформер [33].

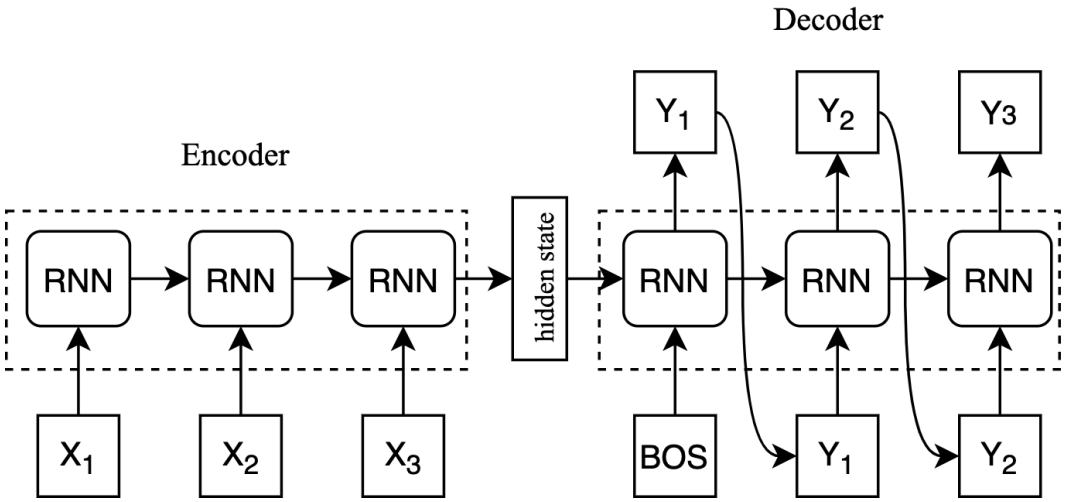


Рисунок 1.5 — Sequence-to-sequence [35] модель для машинного перевода. Перевод последовательности $X_1X_2X_3$ в $Y_1Y_2Y_3$, декодировщик (decoder) использует скрытое состояние кодировщика (encoder) для генерации перевода. Декодировщик последовательно генерирует перевод и использует свои предсказания на каждом следующем шаге генерации. BOS — специальный символ начала декодирования.

1.1.2 Sequence-to-sequence модели и механизм внимания

Разберем механизм внимания на примере sequence-to-sequence модели для машинного перевода (рисунок 1.6), для которой он был изначально предложен в работе «Neural machine translation by jointly learning to align and translate» [37]. С помощью механизма внимая декодировщик получает возможность использовать информацию не только из последнего скрытого состояния кодировщика (hidden state на рисунке 1.6), но и со всех предыдущих. Механизм внимания дает модели более короткий доступ к элементам в любой части входной последовательности.

Вектор c_t, получаемый механизмом внимания, является суммой скрытых состояний кодировщика h_j^{enc} с весами α_{tj}:

$$c_t = \sum_{t=1}^{\text{LEN}(X)} \alpha_{tj} h_j^{enc}.$$

Веса α_{tj} вычисляются с помощью функции softmax по значениям e_{tj}:

$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{\text{LEN}(X)} \exp(e_{tk})},$$

где

$$e_{tj} = a(h_{t-1}^{dec}, h_j^{enc}).$$

Сама функция a, определяющая величину внимания, может быть реализована по-разному:

$$a(x, y) = \begin{cases} v_a^\top \tanh(W_a x + U_a y) & \text{аддитивное внимание [37]} \\ v_a^\top \tanh(W_a [x, y]) & \text{аддитивное [37; 38]} \\ x^\top y & \text{мультипликативное [38]} \\ x^\top W_a y & \text{мультипликативное [38].} \end{cases}$$

Затем, после того как вектор c_t получен для шага декодирования t, он может, например, быть подан на вход декодировщику вместе с результатом декодирования на шаге $t - 1$.

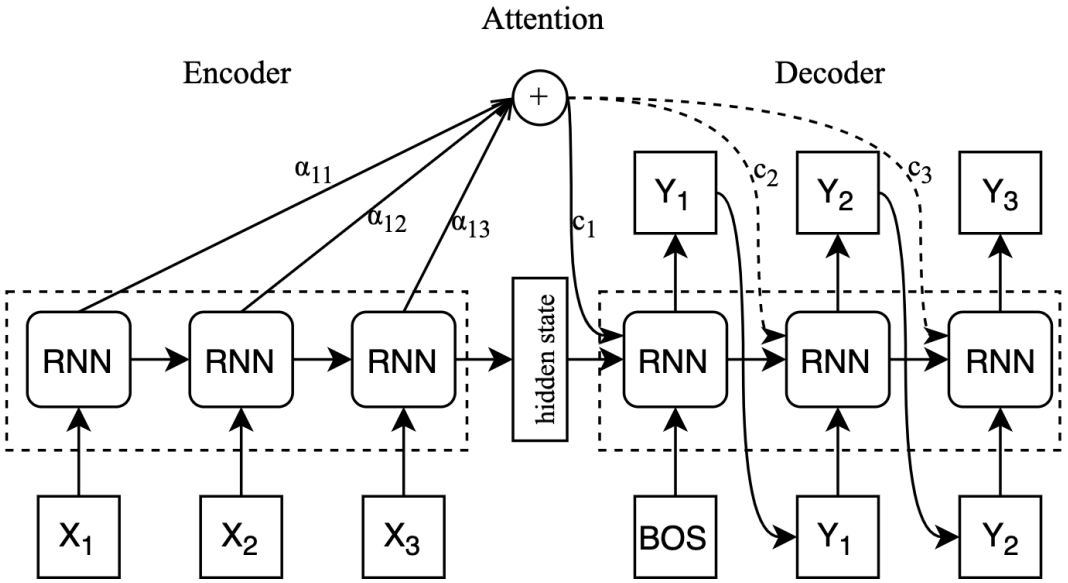


Рисунок 1.6 — Sequence-to-sequence [35] модель для машинного перевода с механизмом внимания [37; 38]. Декодировщик получает средневзвешенные скрытые состояния кодировщика c_t с весами α_{tj} при генерации токена на позиции t с помощью механизма внимания (attention).

1.1.3 Словари и токенизация

В общем случае, на вход языковым моделям подаются не слова, а токены — результат разбиения текста на непересекающиеся подстроки. Например, текст можно разбить на токены по пробелам:

текст	А.Б. Иванов служит в фирме А.Б.В. уже 50 лет, т.е. полвека!
.split()	А.Б.|Иванов|служит|в|фирме|А.Б.В.|уже|50|лет,|т.е.|полвека!

или учесть символы пунктуации:

word_tokenize	А.Б|.|Иванов|служит|в|фирме|А.Б.В|.|уже|50|лет|,|т.е|. |полвека|!
wordpunct_tokenize	А|.|Б|.|Иванов|служит|в|фирме|А|.|Б|.|В|.|уже|50|лет|, |т|.|е|.|полвека|!
razdel.tokenize	А|.|Б|.|Иванов|служит|в|фирме|А|.|Б|.|В|.|уже|50|лет|, |т|.|е|.|полвека|!

Так как текст может быть разбит на токены разными способами (split()[1], word_tokenize[2], wordpunct_tokenize[2], razdel.tokenize[3], spacy[4] и др.), главное использовать один и тот же способ токенизации во время обучения языковой модели, и во время ее применения для других задач.

Словарь языковой модели состоит из токенов, которые встретились в обучающем наборе данных. Для больших наборов данных число уникальных токенов может достигать миллионов, но не все токены будут одинаково часто встречаться в корпусе. По эмпирическому закону Ципфа [39; 40] частота употребления слова в корпусе обратно пропорциональна его рангу, т. е. миллионное слово в словаре встретится примерно 1 раз на 10^6 слов. Поэтому, на практике можно брать такое число слов в словаре, которое позволяет работать с языковой моделью на доступных вычислительных ресурсах. Также, из практических соображений, можно составлять словарь из токенов, покрывающих, например 90%, 95%, 99% всех обучающих данных. Это позволяет найти баланс между размером словаря и его полнотой. На рисунке 1.7 изображен зависимость частоты встречаемости от ранга слова (закон Ципфа) и доля покрытия словаря для набора данных для русского языка, которые были использованы при обучении языковой модели `RuBERT`.

Для словаря токенов любого размера будут возникать ситуации, когда на вход языковой модели попадет текст содержащий новый неизвестный токен (по эмпирическому закону Ципфа распределение частот слов обладает тяжелым хвостом и интегральная сумма не сходится). В таких случаях в словарь добавляют специальный токен [UNK], которым заменяются все неизвестные токены. Такое решение проблемы со словами не из словаря (OOV, out-of-vocabulary) отображает все неизвестные слова в один токен [UNK], хотя все они будут разными и будут нести разную смысловую нагрузку. В слова не из словаря часто попадают имена собственные, названия организаций, узкая терминология, неологизмы, жаргонные слова и те, что не является словами, например, числа, URL-ссылки, элементы программного кода и разметки текста, результаты ошибок в токенизации.

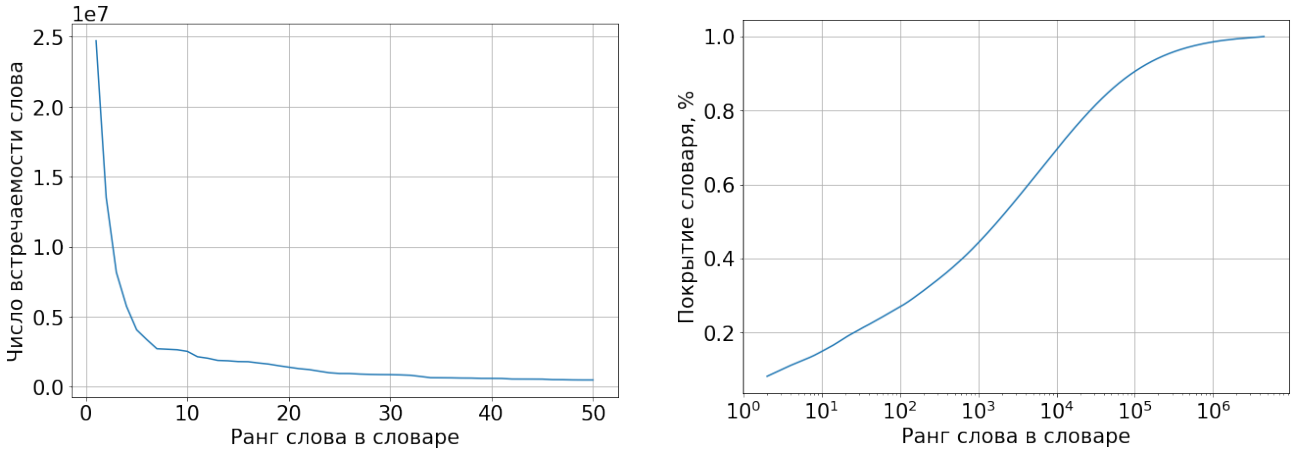
Для редких слов в словаре тоже есть проблема с тем, что языковой модели не хватает данных для того, чтобы выучить хорошие представления и, как

[1] https://docs.python.org/3/library/stdtypes.html#str.split

[2] Токенизатор из nltk.tokenize https://www.nltk.org/api/nltk.tokenize.html

[3] https://github.com/natasha/razdel

[4] https://spacy.io/api/tokenizer



а) Закон Ципфа для первых 50 слов в словаре.

б) Доля покрытия корпуса словарем в зависимости от размера словаря.

Рисунок 1.7 — Закон Ципфа и доля покрытия словаря для набора данных, использованного при обучении языковых моделей для русского языка (`RuBERT`) в рамках работы над диссертацией. Словарь состоит из слов, а не из BPE саб-токенов.

следствие, качественно работать с ними. Эту проблему и проблему со словами не из словаря (OOV) можно решать с помощью построения представления токена из составляющих его символов [41] или N-грамм (например, FastText [4]). В языковой модели, в качестве вектора e_i (рисунок 1.4) будет использоваться объединенный вектор $\overline{e_i}$ полученный из матрицы E и из составляющих символов (или N-грамм) $\overline{e_i} = [e_i, e_i^{char}]$. Вектор e_i^{char} может быть построен с помощью рекурсивных [42], рекуррентных [41; 43] или сверточных сетей [44], работающих на уровне символов или N-грамм.

Словари, состоящие из слов или токенов, всегда будут обладать проблемой того, что не все слова присутствуют в словаре (OOV). В работе «Neural Machine Translation of Rare Words with Subword Units» [45] было предложено разбивать редкие слова и слова не из словаря на части слов (subword units, subtokens, с англ., саб токены, подтокены, подслова или части слов). Единицей словаря становятся саб токены и модель учится работать на уровне саб токенов. Словари саб токенов могут быть построены с помощью разных способов, например, BPE [45; 46], WordPiece [47]. Так как минимальной частью слова является символ и все символы добавляются в такой словарь саб токенов, то любое слово сможет быть разбито на последовательность саб токенов. Пример разбиения на саб токены для модели `RuBERT` (раздел 2.3.2):

слова *языковые* и *трансформер* были разбиты на два сабтокена.

Алгоритм BPE [45; 46] инициализирует словарь сабтокенов символами из обучающего набора данных и специальным символом конца слова. Затем все слова в корпусе разбиваются на последовательности символов и в конец добавляется символ конца слова. Символ конца слова нужен, чтобы можно было восстановить изначальные границы слов. Самая часто встречающаяся пара символов объединяется (merge) и образует новый сабтокен. Разбиение всех слов в корпусе обновляется с учетом нового сабтокена. Затем выбирается новая самая часто встречающаяся пара сабтокенов и снова объединяется. Такую операцию можно повторять до тех пор, пока не будет получен словарь сабтокенов необходимого размера. Таким образом, в результате этого процесса самые частотные слова целиком сами попадают в словарь сабтокенов.

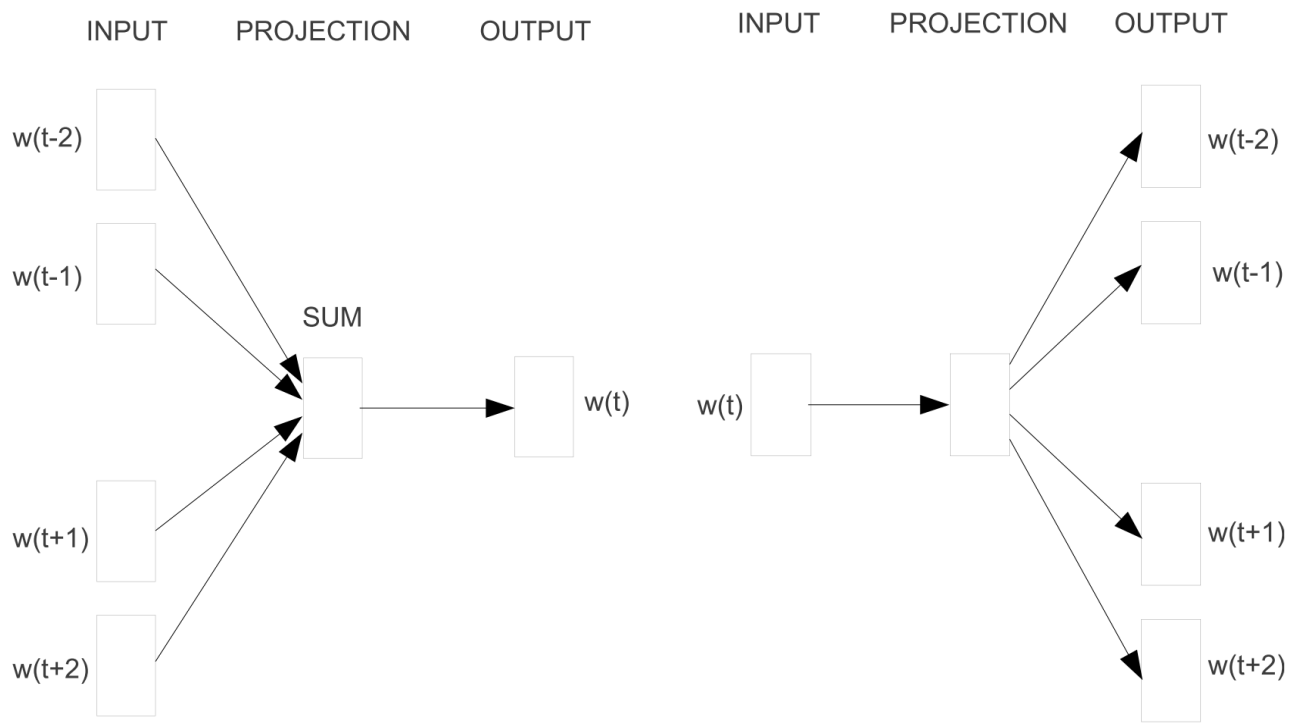
Алгоритм WordPiece устроен похожим образом, только объединяются не самые частотные пары сабтокенов, а максимизирующие правдоподобие униграммной языковой модели [47].

При использовании словарей сабтокенов с полным покрытием [45; 47] общая длина входной последовательности увеличивается по сравнению с использованием словарей на основе слов. Для слов, которые разбиваются на несколько сабтокенов языковая модель не может напрямую предсказать вероятность слова $P(w_N|w_1, w_2, ..., w_{N-1})$, а только последовательно для каждого из его сабтокенов. Если для задачи классификации текстов токенизация играет не такую большую роль, то для, например, задачи разметки последовательностей может потребоваться дополнительная обработка предсказаний модели или данных при работе с сабтокенами.

1.2 Предобучение языковых моделей

Для задач обработки естественного языка в качестве входных признаков обычно используются векторные представления слов word2vec [1; 2], GloVe [3], FastText [4], которые обучены на больших объемах текстовых данных.

Вектора word2vec [1; 2] было предложено обучать двумя способами: CBOW (continuous bag of words) и skip-gram, изображенных на рисунке 1.8. В CBOW вектора слов обучаются быть предсказанными по среднему вектору слов контекста. В skip-gram наоборот, нужно предсказать средний вектор слов контекста по вектору слова. Задача обучения, которая ставится в методе CBOW подходит под обобщенное определение языковых моделей $P(w_i|w_{i-c_L},\ldots,w_{i-1},w_{i+1},\ldots,w_{i+c_R})$ (формула 1.2).



а) CBOW б) Skip-gram

Рисунок 1.8 — Методы CBOW и skip-gram для обучения векторов слов word2vec [1; 2]

При обучении векторов word2vec учитываются только слова, попадающие в контекст ограниченного размера. Метод GloVe (Global Vectors) [3] учитывает статистику совместной встречаемости слов во всем наборе данных и минимизирует функционал J:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^\top \widetilde{w}_j + b_i + \widetilde{b}_j - \log X_{ij})^2,$$

где V — словарь, w_i и $\widetilde{w}_j$ — векторные представлений слов i и j, X — матрица совместной встречаемости слов в корпусе, а функция f определяет вклад частотных и редких сочетаний слов. Для обучения векторов GloVe нужна только матрица совместной встречаемости X и не нужны исходные тексты.

С помощью word2vec и GloVe можно получить вектора только для слов из словаря, также эти методы не учитывают, что у одного слова может быть несколько форм, и обучают вектора для каждой формы слова независимо. FastText [4], позволяет обойти эти ограничения с помощью разбиения слов на символьные N-граммы и обучения векторов для N-грамм. FastText строит вектор для слова как сумму векторов составляющих его N-грамм.

Векторные представления слов word2vec [1; 2], GloVe [3], FastText [4] используются как входные признаки для моделей машинного обучения, но все параметры самой модели обучаются с начальных случайных значений. Предобучение (pre-training) позволяет начать обучения не со случайных значений параметров, а с уже «хороших». Также предобучение позволяет моделям быстрее сходиться и улучшает обобщающие способности [5].

В области компьютерного зрения (computer vision) в 2014 году появились работы в которых начали применять предобучение (pre-training) глубоких нейронных сетей на размеченных данных схожей задачи для дальнейшего дообучения [48] (fine-tuning) или извлечения признаков (feature extraction) [49] уже на целевой задаче. Затем, использование глубоких сверточных нейронных сетей, предобученных на наборе данных ImageNet [50], стало стандартным подходом к решению многих задач компьютерного зрения, как в научной среде [51—55], так и в индустрии.

Для задач обработки естественного языка предобучение нейронных языковых моделей начали использовать в работе «Semi-supervised Sequence Learning» [5]. Относительно небольшая нейросетевая рекуррентная языковая модель с LSTM ячейками [28] (один или два слоя, размерность скрытого слоя 512) предобучалась на целевом наборе данных и потом использовалась для инициализации классификатора. Для классификации использовалось скрытое состояние последнего элемента последовательности. Также в этой работе было показано, что предобучение на неразмеченных данных похожей задачи позволяет повысить качество предсказаний модели. Этот результат схож с тем, что было обнаружено для компьютерного зрения [48].

Для машинного перевода в основном применяются sequence-to-sequence (последовательность в последовательность, кодировщик-декодировщик) [35] модели. Они состоят из двух частей: кодировщик (encoder, энкодер), который строит представление входной последовательности и декодировщик (decoder, декодер), который генерирует перевод по одному слову слева направо. В преды-

дущей работе [5] про предобучение языковых моделей архитектура языковой модели и классификатора были одинаковые, и параметрами языковой модели были инициализированы все параметры модели классификатора (кроме полносвязного классификационного слоя). В работе «Unsupervised Pretraining for Sequence to Sequence Learning» [56] первый рекуррентный слой энкодера и декодера был инициализирован языковыми моделями, матрица векторных представлений слов также инициализировалась матрицами из языковых моделей. Каждая из двух языковых моделей были предобучены на своих языках: исходный и целевой. Второй слой энкодера и декодера инициализировался случайно и обучался только на задаче машинного перевода. Предобученные языковые модели подают на вход второму слою sequence-to-sequence модели уже хорошие векторные представления. Схема модели машинного перевода изображена на рисунке 1.9. Можно сказать, что в этой работе впервые были использованы векторные представления слов из языковых моделей и как источник признаков и для дообучения параметров на целевой задаче.

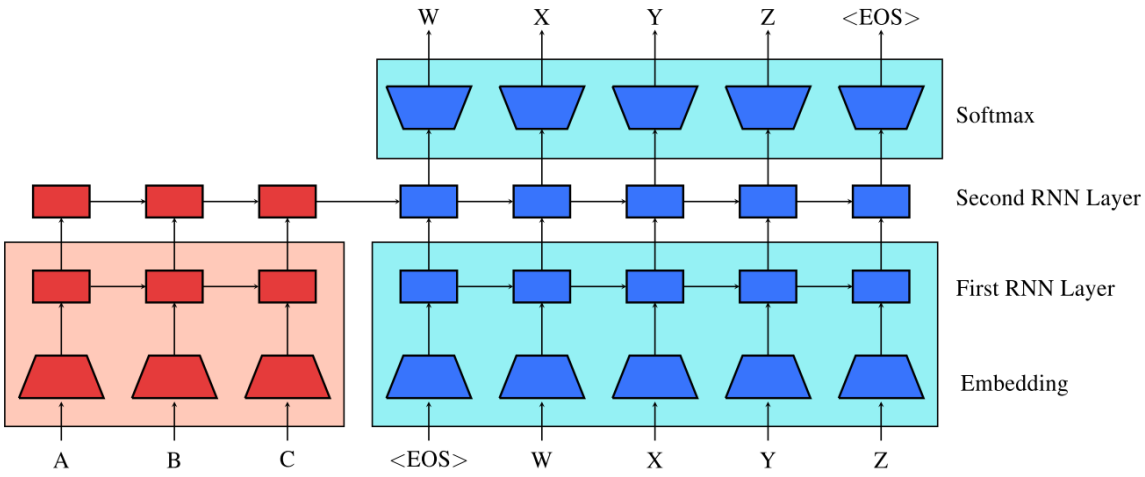


Рисунок 1.9 — Предобучение языковых моделей для машинного перевода. Две языковые для исходного (source) и целевого (target) языков предобучены. Красным цветом выделены параметры энкодера sequence-to-sequence модели машинного перевода, которые инициализированы языковой моделью исходного языка. Синим цветом — параметры декодера, которые инициализированы языковой моделью целевого языка. Рисунок взят из [56].

В качестве задачи для предобучения можно использовать не только языковое моделирование, но и задачи в которых есть достаточно большие наборы данных, например, машинный перевод. В работе «Learned in Translation: Contextualized Word Vectors» [57] был представлен подход CoVe в котором

использовали энкодер sequence-to-sequence модели машинного перевода как источник векторных представлений слов для задач классификации и поиска ответа на вопрос в контексте. Ограничением такого подхода является то, что для предобучения подобных моделей нужны параллельные корпуса для обучения, что затрудняет использование CoVe для языков у которых нет больших параллельных корпусов. Подход CoVe схематически изображен на рисунке 1.10.

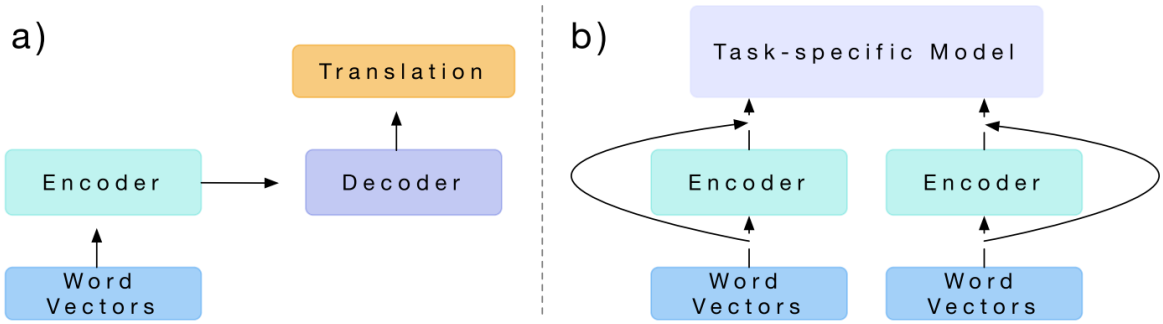


Рисунок 1.10 — CoVe [57]. a) Предобучается энкодер на задаче машинного перевода, b) затем энкодер используется в качестве источника признаков для целевой задачи. Рисунок взят из [57].

Главный прикладной эффект от применения предобученных языковых моделей — это получение качественных векторных представлений слов, которые учитывают контекстные зависимости и позволяют добиваться лучших результатов на целевых задачах. Такого прикладного эффекта можно добиваться не только просто предобучением одной языковой модели. В работе «Semi-supervised sequence tagging with bidirectional language models» [58] предлагают использовать объединенный вектор-признак от двух языковых моделей (Bi-LM): одной, которая была обучена предсказывать следующее слово при проходе последовательности справа налево и второй — слева направо.

Затем, в последующей работе был предложен метод `ELMo` [6] (Embeddings from Language Models) для более эффективного использования скрытых состояний многослойных языковых моделей, как контекстно-зависимых векторных представлений слов. В ней также используются две рекуррентные языковые модели (Bi-LM) работающие в двух противоположных направлениях, но в отличии от предыдущей работы у этих двух языковых моделей общие параметры матрицы векторных представлений слов и выходного softmax слоя. Идея `ELMo` заключается в том, что для разных задач могут быть важны признаки с разных слоев языковых моделей и вклад каждого слоя в векторное представление

можно учить вместе с задачей:

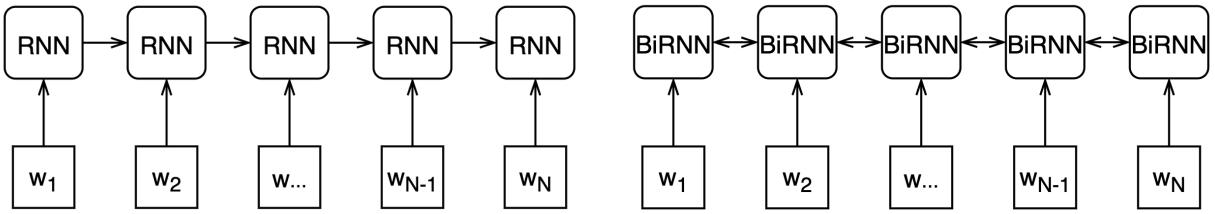
$$ELMo(t, \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} H_{jt},$$

где $H_{jt} = [\overrightarrow{H_{jt}}, \overleftarrow{H_{jt}}]$ — конкатенация скрытых состояний слоя j языковых моделей, Θ^{task} — параметры γ^{task} и s_j^{task} линейной комбинации слоев, которые учатся под задачу, сумма s_j^{task} равна 1. Нулевой слой — векторные представления слов, $1 \dots L$ — слои рекуррентной языковой модели. Такой способ позволил достичь новых лучших результатов на многих задачах: классификация, разметка последовательностей, поиск ответа на вопрос, разрешение кореференции.

В работе «Universal Language Model Fine-tuning for Text Classification» [59] был предложен трехэтапный метод для обучения модели на целевой задаче. Первый этап — обучение языковой модели на большом наборе текстов общего домена. Второй этап — дообучение языковой модели на текстах из домена целевой задачи, например, на самих данных целевой задачи. Третий — обучение всей архитектуры на целевой задаче. Каждый из этих трех этапов был известен ранее, но предложенная комбинация и примененные техники обучения, такие как триангулярное изменение скорости обучения (triangular learning rate schedule), постепенное размораживание слоев (gradual unfreezing), использование объединения вектора последнего скрытого состояния рекуррентной сети и операций субдискретизации (max-, avg- pooling) оказались очень эффективны на практике.

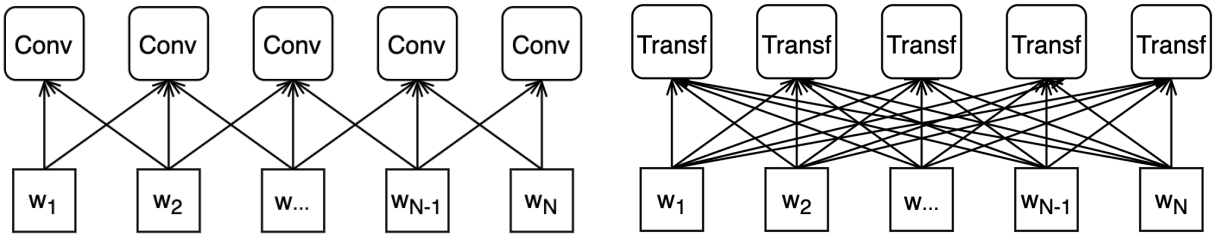
Предыдущие работы использовали языковые модели основанные на рекуррентных или сверточных сетях. В 2017 году в работе «Attention is All You Need» [33] была представлена архитектура Трансформер состоящая из полносвязных слоев и механизма внимания. Подробнее про Трансформер написано в разделе 2.1. Работа с последовательностями слов рекуррентной, сверточной и Трансформер сети изображены на рисунке 1.11. Рекуррентные сети обрабатывают входные данные последовательно, сверточные — параллельно, но имеют доступ только к нескольким ближайшим словам (задается размером свертки), Трансформер — параллельно, имеют доступ ко всем словам в последовательности.

На базе архитектуры Трансформер в 2018 году появилась модель GPT (generative pre-training, генеративное предобучение) [7]. GPT — предобученная



а) Рекуррентная сеть.

б) Двунаправленная рекуррентная сеть.



в) Сверточная сеть.

г) Трансформер сеть.

Рисунок 1.11 — Порядок обработки входных данных разными архитектурами нейронных сетей, работающих с последовательностями слов.

языковая модель состоящая из 12 Трансформер слоев (около 110 миллионов параметров). Механизм внимания в Трансформер слое работает со всей входной последовательностью, но для задачи языкового моделирования нельзя заглядывать в будущее — нельзя смотреть на слова, которые идут после того, для которого делается предсказание следующего слова. Для этого в модели GPT был использован механизм внимания с маской (masked self-attention). Маска позволяет механизму внимания смотреть только на текущий и предыдущие слова и скрывает все последующие слова. На рисунке 1.12 изображен механизм внимания (self-attention) и механизм внимания с маской (masked self-attention) для языкового моделирования.

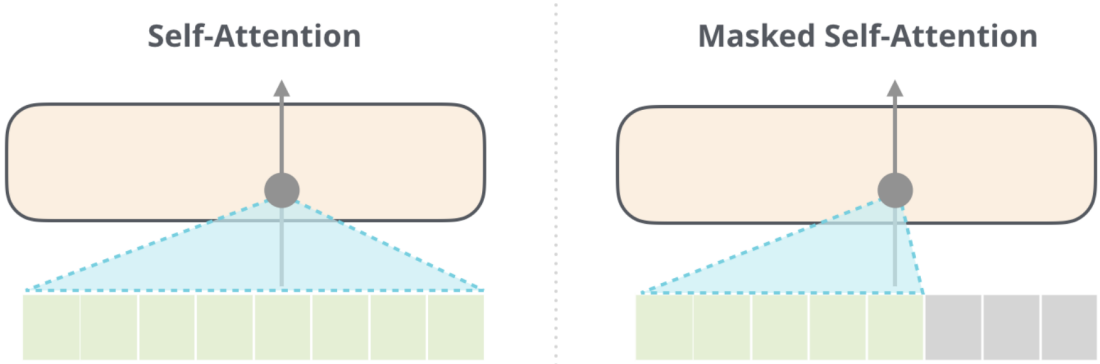


Рисунок 1.12 — Механизм внимания (self-attention) и механизм внимания с маской (masked self-attention). Рисунок взят из http://jalammar.github.io/illustrated-gpt2/.

Для применения модели GPT [7] к задачам обработки естественного языка используется специальный символ *Extract*, который приписывается к входному тексту и затем к выходу с последнего слоя для этого символа применяется линейный полносвязный слой. Этот слой уже обучается решать целевую задачу. Подробнее изображено на рисунке 1.13.

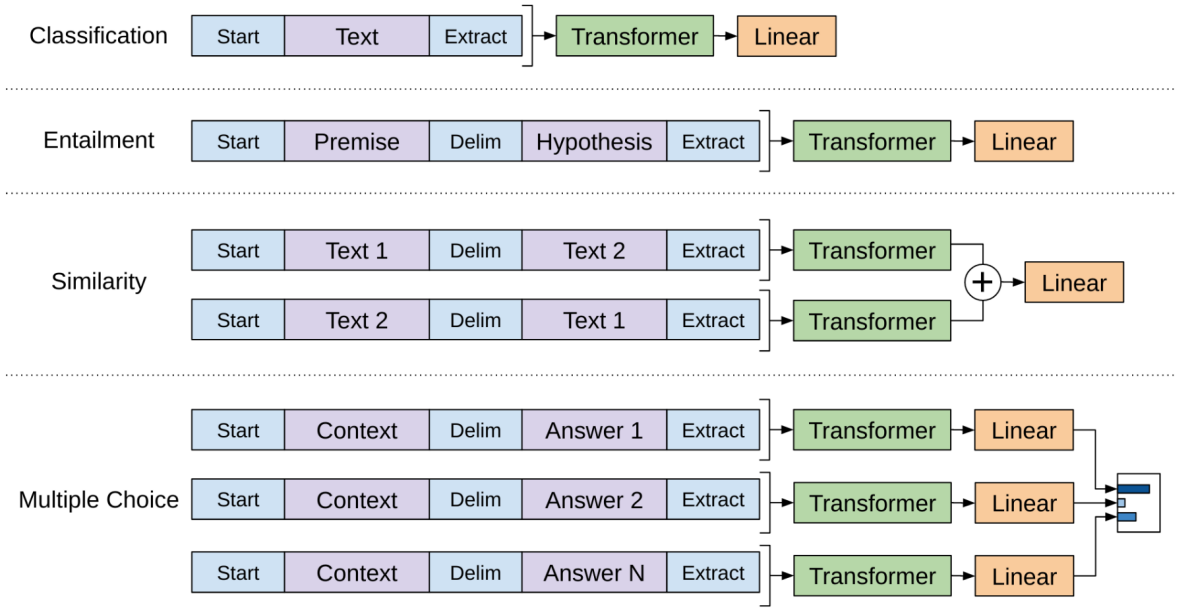


Рисунок 1.13 — Применение предобученной языковой модели GPT [7] к разным текстовым задачам. В модели GPT используются специальные символы для обозначения начала последовательности (Start), границы между текстами (Delim) и символ Extract, к которому потом применяется линейный слой для решения целевой задачи.

Модели BERT [8] также как и GPT состоят из Трансформер слоев. Они немного отличаются форматом входных данных (BERT использует другой набор специальных символов) и задачами предобучения. Если GPT обучается задаче языкового моделирования, то BERT обучается на двух задачах: маскированное языковое моделирование (MLM) и предсказание следующего предложения (NSP). Два варианта моделей BERT было опробовано авторами: BERT-Base — 12-слойный Трансформер (110 миллионов параметров, как GPT) и BERT-Large — 24-слойный Трансформер (340 миллионов параметров). BERT-Base улучшил результаты модели GPT на текстовых задачах, а увеличение размера предобученной модели (BERT-Large) показал еще более высокие результаты. Более подробно про подход к предобучению BERT и про модели BERT описан в разделе 2.2.

Появление моделей BERT дало старт активному развитию и возникновению различных предобученных языковых моделей. Подробный обзор предобученных моделей и их систематизация произведены в работе [60]. В таблице 1 приведены только некоторые из них таких моделей.

Таблица 1 — Модели и походы появившиеся как развитие моделей BERT [8], GPT [7].

Модель	Описание подхода
XL-Net [61]	Максимальная длина последовательности у моделей BERT ограничена значением в 512 токенов. XL-Net основан на Transformer-XL [62], в котором добавлена рекуррентность и позиции токенов кодируются с помощью относительных, а не абсолютных значений. Кроме этого в XL-Net скрытые токены для задачи MLM предсказываются авторегрессионно, а не одновременно и независимо как в BERT.
RoBERTa [63]	Улучшенный процесс обучения по сравнению с BERT: увеличен размер батча (8000), маски для задачи генерируются случайно для каждой эпохи, не используют предсказание следующего предложения.
ALBERT [64]	Используются Трансформер слои с общими параметрами (Universal Transformers [65]). Это позволяет уменьшить число обучаемых параметров и предобучать более глубокие языковые модели, так как число параметров модели перестает зависить от числа слоев.
BART [66]	Подавляющий шум sequence-to-sequence автокодировщик (denoising autoencoder) с Трансформер слоями, учится авторегрессионно восстанавливать входную последовательность в которой случайные слова и фрагменты текста заменяются на маску или выкидываются, изменен порядок предложений. Лучше чем BERT работает на генеративных задачах.

ELECTRA [67]	Для предобучения используют задачу определения заменённого токена RTD (replaced token detection) вместо MLM. Таким образом, вместо замены 15% токенов на специальный токен-маску, 15% случайных токенов заменяются на предсказания модели обученной на MLM задаче. Модель ELECTRA (дискриминатор) учится определять какие из токенов в тексте были изменены на предсказания модели, обученной на MLM (генератор). Это позволяет более быстро и эффективно обучать модель, так как обучающий сигнал идет от всех токенов, а не только от 15% как в модели `BERT`.
GPT-2 [9] и GPT-3 [10]	Развитие модели GPT. Обучены модели GPT-2 разных размеров от 117 миллионов параметров (примерно соответствует `BERT-Base`) до 1,5 миллиардов. GPT-3 — 175 миллиардов параметров. С увеличением размера модели и увеличением объема данных для обучения возрастает качество генерируемых текстов и способность модели решать целевую задачу на меньшем числе примеров: от нуля (zero-shot) до единиц или десятков примеров (few-shot). Для модели GPT-3 была обнаружена способность обучаться задаче не применяя алгоритм обратного распространения ошибки. Например, можно сформировать входной текст с несколькими примерами выполнения задачи вначале и для решения задачи для нового примера запустить генерацию продолжения текста: *Translate English to French: sea otter => loutre de mer; cheese => ...*

1.3 Применение предобученных векторных представлений слов

Для решения задач обработки естественного языка методами машинного обучения необходимо представить текст в виде одного или нескольких векторов признаков. Текст может состоять из слов, чисел, пунктуации, специальных символов дополнительной разметки (например, HTML). Каждую такую «единицу» можно представить в виде вектора разными способами, например, с помощью унитарных кодов (one-hot encoding) и контекстно-независимых векторных пред-

ставлений Word2Vec [1; 2], GloVe [3], FastText [4]. Предобученные языковые модели также могут служить источниками контекстно-зависимых векторных представлений слов для последующего их использования в качестве признаков для моделей машинного обучения.

1.3.1 Контекстно-независимые векторные представления слов

Для каждого слова w из словаря, контекстно-независимые предобученные векторные представления слов ставят в соответствие вектор e_w из матрицы E. На рисунке 1.14 изображено как предобученные вектора подаются на вход модели машинного обучения. Во время тренировки модели машинного обучения матрица векторных представлений E может быть как зафиксирована (freezed), так и быть дообучена (fine-tuned) на целевой задаче.

Рисунок 1.14 — Векторные представления слов как признаки для моделей машинного обучения.

1.3.2 Контекстно-зависимые векторные представления слов

Выходы с последнего слоя языковых моделей могут быть использованы как векторные представления слов (рисунок 1.15). Такие векторные представления слов являются контекстно-зависимыми, т. е. формируемый вектор учитывает окружающие слова, чего не происходит с контекстно-независимыми моделями Word2Vec [1; 2], GloVe [3], FastText [4]. Учитывать контекст важно для задач обработки текста, так как некоторые слова могут нести разный смысл в зависимости от контекста:

Жизнь бьет **ключом**.
Отрыть дверь **ключом**.

В этих двух предложениях вектора e_w для слова **ключом** будут одинаковыми, но вектора h_w, полученные из языковой модели, будут отличаться и отражать разницу в значениях омонимичного слова **ключ**.

Языковая модель вместе с матрицей векторных представлений слов может быть как зафиксирована (freezed) или быть дообучена (fine-tuned) на целевой задаче. Например, в работе «To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks» [68] были проанализированы эти две стратегии для моделей `ELMo` и `BERT` и было показано, что модель `ELMo` лучше оставлять зафиксированной, а модель `BERT` дообучать на целевой задаче. Также было показано, что перед тем, как обучать уже предобученные языковые модели на целевой задаче, их можно еще предобучить на наборе релевантных задач [69].

В этой главе было введено понятие языковой модели и его обобщения для маскированного языкового моделирования, произведен обзор методов предобучения языковых моделей и применения контекстно-независимых и контекстно-зависимых векторных представлений слов.

В главе 2 будет описана процедура предобучения языковых моделей `BERT` и метод переноса знаний.

Применение обученных языковых моделей `BERT` к различным задачам обработки естественного языка и анализ результатов представлен в главах 3, 4, 5.

Рисунок 1.15 — Векторные представления слов из языковых моделей как признаки для моделей машинного обучения.

Глава 2. Обучение языковых моделей на базе архитектуры Трансформер

В предыдущей главе был изложен процесс развития языковых моделей. На данный момент, языковые модели на базе архитектуры Трансформер [33] показывают наилучшие результаты на многих задачах обработки естественного языка [7; 8; 10; 61; 63; 64; 66; 67]. Поэтому основным предметом исследования данной работы являются языковые модели на основе архитектуры Трансфомер и, в частности, модель `BERT` [8].

Сначала в разделе 2.1 будет описана архитектура Трансформер, затем в разделе 2.2 будет описан метод предобучения языковых моделей `BERT`, и в разделе 2.3 будет представлен предложенный метод переноса знаний с обученных моделей `BERT` на другие языки и домены.

2.1 Архитектура Трансформер

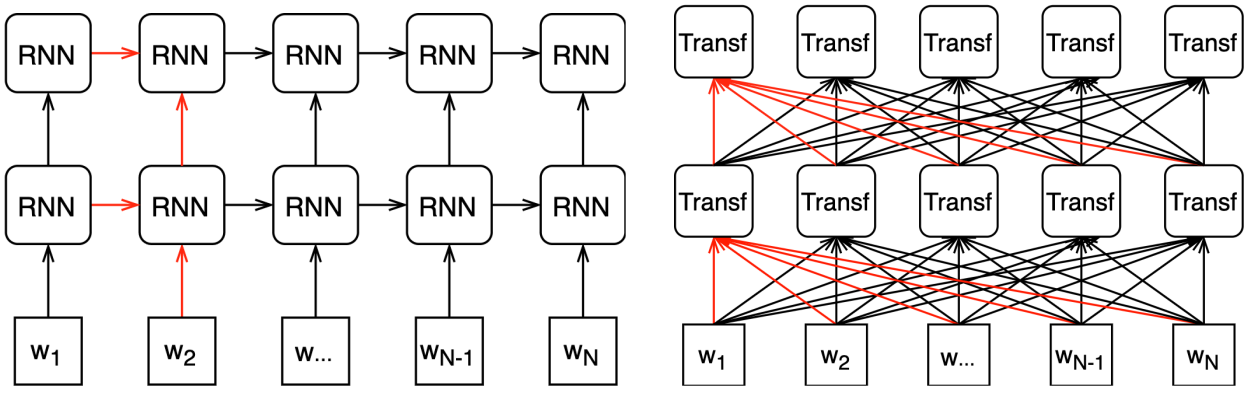
Архитектура Трансформер была предложена в 2017 году в работе «Attention Is All You Need» [33], где она была применена к задаче машинного перевода. В машинном переводе используют sequence-to-sequence [35] модели, состоящие из кодировщика (encoder) и декодировщика (decoder). Кодировщик обрабатывает текст на исходном языке (source), а декодер, используя информацию от кодировщика, генерирует перевод на целевой язык (target). Кодировщик и декодировщик могут быть реализованы с помощью рекуррентных [35] или сверточных сетей [36]. В работе «Attention Is All You Need» [33] вместо них была использована архитектура Трансформер. Подробнее про sequence-to-sequence модели описано в разделе 1.1.1.

Архитектура Трансформер состоит из полносвязных слоев и механизма внимания (attention) [37; 38]. Ранее, механизм внимания применялся в sequence-to-sequence [35] моделях декодировщиком для получения релевантной на данном шаге декодирования информации c_t из кодировщика [37; 38]. Подробнее про механизм внимания рассказано в разделе 1.1.2.

2.1.1 Self-attention

Изначально, механизм внимания был предложен для передачи информации от кодировщика декодировщику в sequence-to-sequence моделях [37; 38]. Механизм внимания, примененный к самой входной последовательности называется self-attention (внимание на себя). Термин был впервые употреблен в работе «A structured self-attentive sentence embedding» [70] при построении векторного представления предложения для задач текстовой классификации. Это векторное представление ранее строилось либо из последнего скрытого состояние рекуррентной сети либо с помощью операций субдискретизаций (max-pooling, average-pooling), либо объединением перечисленных трех представлений [59].

В архитектуре Трансформер механизм внимания используется между слоями для передачи информации от элементов входной последовательности с предыдущего слоя на следующий. Механизм внимания применяется последовательно от слоя к слою к самой входной последовательности для каждого ее элемента, схематически это изображено на рисунке 2.1. Красным цветом выделено как передается информация в двухслойной рекуррентной сети и с помощью механизма self-attention в двухслойном Трансформере.



а) Двухслойная рекуррентная сеть. б) Двухслойный Трансформер.

Рисунок 2.1 — Передача информации в рекуррентной и Трансформер сетях.

2.1.2 Трансформер

Как уже было сказано ранее, архитектура Трансформер состоит из повторяющихся полносвязных слоев и механизмов внимания, образующих Трансформер слои, из которых в свою очередь состоят кодировщик и декодировщик (рисунок 2.2). Это первая модель, которая позволяет работать с последовательностями только за счет механизма внимания, без рекуррентности и сверток.

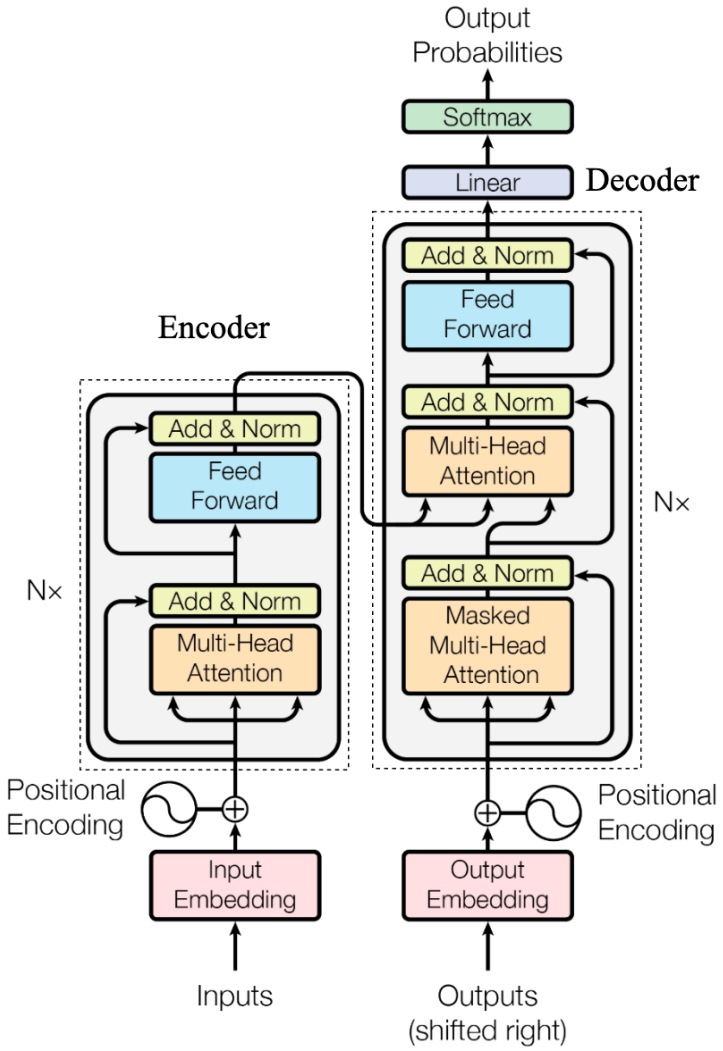


Рисунок 2.2 — Архитектура Трансформер [33]. Кодировщик состоит из N повторяющихся слоев, состоящих из механизма внимания multi-head-self-attention и полносвязного слоя. Декодировщие состоит из N повторяющихся слоев, состоящих из механизма внимания multi-head-self-attention, механизма внимания на последний слой кодировщика и полносвязного слоя.

Кодировщик (encoder на рисунке 2.2) состоит из N повторяющихся Трансформер слоев. Трансформер слой кодировщика состоит из:

1. механизм внимания, а именно multi-head-self-attention — вариант механизма внимания, примененного к самой последовательности (self-attention), которые формирует не один, а несколько векторов внимания (heads);
2. полносвязный слой, применяемый к каждому элементу последовательности независимо.

Также используются residual (остаточные) [71] связи вокруг полносвязного слоя и механизма внимания, и нормализация слоя (layer normalization) [72]: $\text{LayerNorm}(x + \text{SubLayer}(x))$, где SubLayer — механизм внимания либо полносвязный слой.

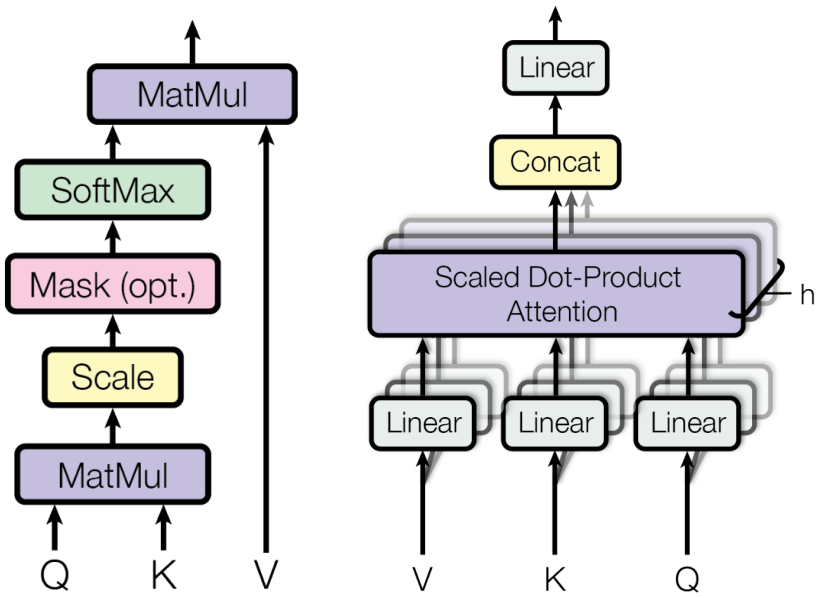
Декодировщик (decoder на рисунке 2.2) также состоит из N повторяющихся слоев. В Трансформер слой декодировщика, в дополнение к двум составляющим, еще добавляется механизм внимания на выход из последнего слоя кодировщика. Так как декодировщик предсказывает следующее слово $i+1$ по предыдущим, то при обучении ему нельзя использовать информацию о будущих словах. Для этого в механизме внимания зануляется вес слов, следующих за словом в позиции i (masked self-attention на рисунке 1.12).

В архитектуре Трансфомер механизм внимания рассматривается как механизм извлечения значений V (values) по ключам K (key) на запросы Q (query). Механизм внимания формирует вектор, который является взвешенной суммой значений V. Для получения весов для взвешенной суммы значений V $(n_k \times d_v)$ к скалярному произведению между запросами Q $(n_q \times d_k)$ и ключами K $(n_k \times d_k)$ применяется функция softmax:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V, \tag{2.1}$$

где n_q, n_k — число запросов и ключей в матрице запросов Q и ключей K соответственно $(n_v = n_q)$, $d_q = d_k$, d_k, d_v — размерности запросов, ключей и значений. Взвешенная сумма получается в результате произведения вектора весов (выход softmax) и матрицы значений V. Произведение $QK^\top$ масштабируется на значение $\frac{1}{\sqrt{d_k}}$. Вычисление произведения двух матриц $QK^\top$ имеет вычислительную сложность $O(n_k \times n_q)$ и в случае self-attention $(n_k = n_q = n)$ является квадратичной от длины последовательности n.

Механизм внимания архитектуры Трансформер изображен на рисунке 2.3a.



а) Q,K,V–внимание

б) Многоголовое внимание (multi-head attention)

Рисунок 2.3 — Механизм внимания в архитектуре Трансформер.

Механизм внимания (формула 2.1) для каждого запроса из Q возвращает один вектор внимания. Авторы «Attention Is All You Need» [33] предложили возвращать h независимых векторов внимания (heads, голов) за счет обучения h независимых линейных преобразований над матрицами Q, K и V. Затем, к ним применяется h механизмов внимания (формула 2.1) и их результат объединяется в один вектор:

$$\text{MultiHead}(Q, K, V) = [head_1, head_2, \ldots, head_h]W^O,$$

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V),$$

где W_i^Q $(d \times d_k)$, W_i^K $(d \times d_k)$, W_i^V $(d \times d_v)$ — матрицы линейных преобразований, d — размерность запросов, ключей и значений. Такой механизм внимания, состоящий из нескольких голов (heads), называется multi-head-attention (многоголовое внимание) и изображен на рисунке 2.3б. Число голов обычно выбирается в зависимости от числа слоев — чем больше слоев, тем больше голов внимания, например, 12 слоев 8 голов [33], 12 слоев 12 голов [8], 24 слоя 16 голов [8], 96 слоев 96 голов [10].

Архитектура Трансформер никак не использует порядок элементов входной последовательности. Для каждого элемента последовательности все операции в Трансформер слое (внимание, применение полносвязного слоя)

происходят независимо для разных элементов последовательности. Чтобы добавить информацию об абсолютном или относительном положении во входной последовательности используются векторные представления позиций (position embeddings). Векторные представления позиций добавляются к входным векторным представлениям. Вектора для позиций могут быть обучены вместе с параметрами всей модели [8; 36] или могут быть зафиксированы и заданы аналитически [33].

2.2 Предобучение языковых моделей BERT

Под названием BERT понимают как метод предобучения языковых моделей с архитектурой Трансформер, так и обученные этим методом модели. В данном разделе сначала будет описан метод BERT предобучения языковых моделей с архитектурой Трансформер, а затем в разделе 2.2.5 будут приведены примеры конкретных моделей BERT.

Ключевые особенности метода предобучения BERT:

– Обработка всей последовательности одновременно. Авторы называют такую обработку двунаправленной (bidirectional) [8]. Стоит различать двунаправленность в рекуррентных сетях и в сетях с архитектурой Трансформер. В двунаправленных рекуррентных сетях входные данные обрабатываются последовательно по одному токену слева направо и справа налево. В сетях с архитектурой Трансформер, каждый токен обрабатывается параллельно и имеет доступ с помощью механизма внимания ко всем остальным токенам. В некотором смысле, в этом случае нет направления обработки данных, а термин двунаправленность возникает от того, что есть доступ к контексту и слева и справа одновременно.

– Обучение без учителя (unsupervised learning), в том смысле что для обучения не нужны предварительно размеченные данные. Термин self-supervised learning больше подходит для описания метода предобучения BERT, так размеченные данные для обучения можно получить из самих текстов, например, предсказание следующего слова — эта разметка уже сама по себе есть в данных. Подобные задачи, данные для обучения

которым уже представлены в исходных данных, еще называют вспомогательными задачами (auxilary tasks).

2.2.1 Сравнение BERT с ELMo, GPT

В модели BERT представления для токенов на слое N строятся используя представления всех токенов на слое $N-1$, т. е. для токена в позиции i доступны все токены левого контекста $[1,2,\ldots,i-1]$ и все токены правого контекста $[i+1,i+2,\ldots,L]$. Авторы модели BERT называют это двунаправленностью.

Двунаправленность отличает модель BERT от моделей GPT и ELMo (Рисунок 2.4). В GPT используется только левый контекст $[1,2,\ldots,i-1]$ для построения представлений токенов. В ELMo используются представления с двух независимых рекуррентных сетей: одна обрабатывает последовательность слева направо, вторая — справа налево. В ELMo тоже есть двунаправленность, но представления с каждого из направлений получаются независимыми.

Модели BERT и GPT не отличаются архитектурно, а только способом обработки входных данных. Обе модели состоят из Трансформер слоев.

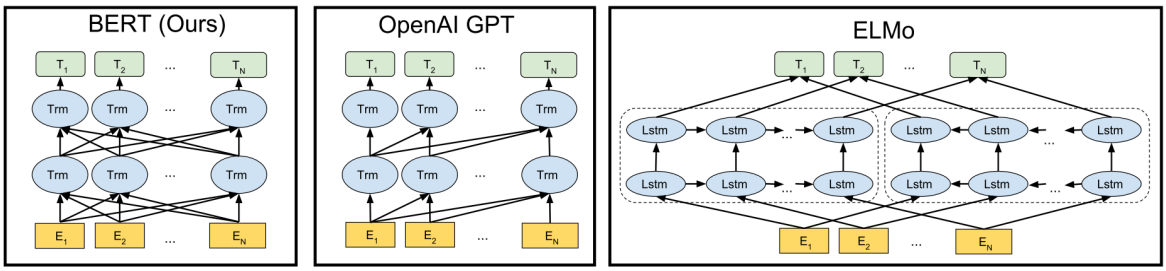


Рисунок 2.4 — Сравнение архитектур BERT [8], GPT [7], ELMo [6].

2.2.2 Задачи предобучения

Модель BERT обучается на двух задачах:

– MLM — маскированное языковое моделирование (masked language modeling),

– NSP — предсказание следующего предложения (next sentence prediction),

и общая функция потерь является суммой функций потерь для каждой из задач.

Задача маскированного языкового моделирования (masked language modeling) состоит в том, чтобы предсказать какие токены в исходной последовательности были заменены случайным образом на специальный токен-маску [MASK]:

Язык ##овые модели на базе архитектуры трансформ ##ер
MLM:
Язык [MASK] модели на [MASK] архитектуры трансформ ##ер

Во время обучения моделей `BERT` 15% случайных токенов заменяется:

– в 80% из 15% на токен-маску [MASK].
– в 10% заменяется на случайное слово.
– в 10% оставляется неизмененным.

Замена на токен [MASK] происходит не всегда для того, чтобы уменьшить разницу между входными данными на этапе предобучения модели и на этапе использования модели на каких-либо задачах обработки текстов, в которых токен-маска [MASK] не встретится. Модель учится предсказывать только эти 15% токенов, в отличии от обычных языковых моделей, которые предсказывают все токены в последовательности по очереди.

В последующих обновлениях подхода к предобучению `BERT` было добавлено маскирование слова целиком (whole-word masking)[1]. В задаче MLM может быть скрыт любой случайный токен. Слова часто разбиваются на несколько токенов и если скрыт только один из них, то восстановить такой токен оказывается проще, так как уже известны другие токены из этого слова. Чтобы сделать обучение более эффективным, задача MLM была несколько усложнена (изменяются 15% слов, а не токенов):

Язык ##овые модели на базе архитектуры трансформ ##ер
MLM:

[1]Обновление от 31 мая 2019 https://github.com/google-research/bert/blob/master/README.md

Язык [MASK] модели на [MASK] архитектуры трансформ ##ер
MLM и маскирование слова целиком:
[MASK] [MASK] модели на [MASK] архитектуры трансформ ##ер

В таблице 2 приведены результаты с добавлением маскирования слова целиком и без. Маскирование слов целиком не было использовано в диссертационной работе, так как появилось уже после того, как все эксперименты были проведены.

Таблица 2 — Модели `BERT` с маскированием слова целиком (whole-word masking, WWM в таблице). Результаты на наборах данных для ответа на вопросы в контексте SQuAD [73] и наборе данных Multi NLI [74]

Модель	SQuAD 1.1 F_1/EM	Multi NLI Accuracy
BERT-Large, Uncased	91.0/84.3	86.05
BERT-Large, Uncased (WWM)	**92.8/86.7**	**87.07**
BERT-Large, Cased	91.5/84.8	86.09
BERT-Large, Cased (WWM)	**92.9/86.7**	**86.46**

Задача предсказания следующего предложения (next sentence prediction) состоит в том, что модель `BERT` обучается определять являются ли два предложения (фрагмента текста) идущими подряд в исходном тексте. Такая задача заставляет модель `BERT` строить отношения между предложениями, чего нет в обычных языковых моделях в явном виде. Предсказание следующего предложения — это задача бинарной классификации, положительные примеры берутся из исходного текста, отрицательные выбираются случайным образом из всего обучающего набора данных. Для предсказания $P(is\_next\_sentence)$ следующего предложения используется специальный токен [CLS] (рисунок 2.5), выход с последнего слоя N модели $BERT^N_{[CLS]}$ для этого токена подается в полносвязный слой FC_1 с функцией активации $tanh$ размерности 768 (для `BERT-Base`, 1024 для `BERT-Large`) и затем в линейный полносвязный слой FC_2 с функцией softmax и размерностью выхода 2:

$$P(is\_next\_sentence) = softmax(FC_2(tanh(FC_1(BERT^N_{[CLS]})))).$$

После предобучения выход из $tanh(FC_1(BERT^N_{[CLS]}))$ также используется для применения моделей `BERT` к задачам классификации.

2.2.3 Формат входных данных

Модель BERT использует служебные токены [MASK] — для маскирования токена, [SEP] – для обозначения конца предложений (фрагментов текста), [CLS] – специальный токен, который используется для задачи предсказания следующего предложения и задач классификации.

Входной текст сначала разбивается на токены с помощью алгоритма BPE (имплементация WordPiece [47]), в самое начало входной последовательности вставляется специальный токен [CLS], затем идут токены первого фрагмента текста, затем токен разделитель [SEP], затем токены второго фрагмента текста и в конце еще один токен разделитель [SEP]. Формат входных данных изображен на рисунке 2.5.

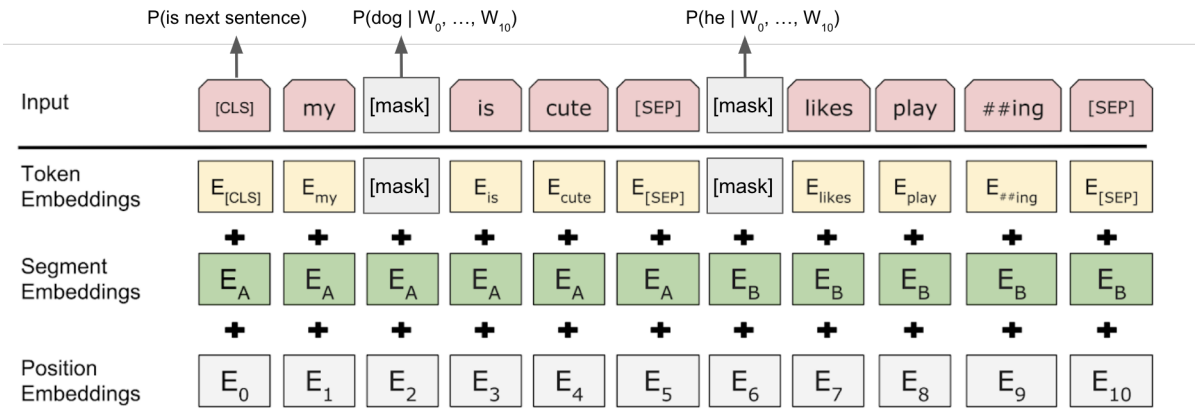


Рисунок 2.5 — Формат входных данных для модели BERT и формирование входных векторных представлений.

Входные векторные представления состоят из трех компонент:

- Векторное представление токена (token embeddings) — каждому токену из словаря соответствует свой вектор.
- Векторное представление сегмента (segment embeddings) — всего два вектора для каждого из сегментов (первый и второй фрагмент текста). Позволяет дополнительно указать модели BERT к какому предложению принадлежит токен.
- Векторное представление позиции в тексте (position embeddings) — вектор, кодирующий информацию о номере токена во входной последовательности.

Итоговый вектор является суммой из трех векторов перечисленных типов (рисунок 2.5), каждый из типов векторов обучается вместе со всей моделью BERT.

2.2.4 Особенность BERT как языковой модели

В литературе языковую модель зачастую определяют как модель, описывающую вероятность $P(w_1, w_2, \ldots, w_N)$ встретить последовательность слов $w_1, w_2, \ldots, w_N$ (подробнее про языковые модели в разделе 1.1). По определению условной вероятности (chain rule) можно расписать:

$$P(w_1, w_2, \ldots, w_N) = P(w_1)P(w_2|w_1) \ldots P(w_N|w_1,w_2, \ldots ,w_{N-1}).$$

Модель BERT была обучена моделировать распределение $P(w_i|w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_N)$ по словарю для пропущенного слова (токена) в контексте. Для каждого слова w_i нужно знать не только левый, но и правый контекст. Вероятности $P(w_1, w_2, \ldots, w_N)$ (или хотя бы $P(w_i|w_1, \ldots ,w_{i-1})$) не выражаются через вероятности вида $P(w_i|w_1, \ldots ,w_{i-1},w_{i+1}, \ldots ,w_N)$. Поэтому модель BERT неправильно называть языковой моделью в том смысле, что она не моделирует распределение $P(w_1, w_2, \ldots, w_N)$. Авторами BERT также был дан комментарий по поводу того является ли BERT языковой моделью[2]. Также были попытки представить BERT как языковую модель, основанную на марковской сети (MRF-LM [75]) в работе [76], но позже в ней была найдена ошибка[3].

В разделе 1.1 диссертации введено обобщение понятия языковой модели (формула 1.2) в соответствии с которой модель BERT является маскированной языковой моделью.

Стоит отметить, что приведенные выше нюансы определения BERT, как языковой модели, не мешают использовать ее для формирования эффективных контекстно-зависимых представлений слов.

[2] https://github.com/google-research/bert/issues/35#issuecomment-435444962

[3] https://sites.google.com/site/deepernn/home/blog/amistakeinwangchoberthasamouthanditmusts

2.2.5 Предобученные модели BERT

Изначально авторами подхода BERT к предобучению языковых моделей были опубликованы следующие модели для английского языка:

- BERT-Base, 12-слойный Трансформер, 768 размер скрытого слоя, 12 голов внимания (attention heads), 110 миллионов параметров;
- BERT-Large, 24-слойный Трансформер, 1024 размер скрытого слоя, 16 голов внимания (attention heads), 340 миллионов параметров.

Два варианта для каждой модели были обучены на текстах полностью приведенных в нижний регистр (Uncased) и с сохранением оригинального регистра (Cased). Base и Large характеризуют размер модели.

Позже, была выложена многоязычная BERT-Base модель обученная на 104 языках и данных из Википедии и модель BERT-Base для китайского языка[4]. Появление многоязычной BERT-Base модели открыло возможность использовать модель BERT для остальных языков и дало ход многим работам по исследованию переносимости знаний с одного языка на другой, обучению без примеров (zero-shot learning) и обучению на малом числе примеров (few-shot learning) [77—80]. Многоязычная BERT-Base модель используется в этой работе для обучения языкоспецифичных моделей RuBERT и славянский BERT (см. раздел 2.3.2).

На данный момент большое число предобученных языковых моделей (не только BERT) доступны в каталоге моделей[5] для использования с репозиторием Transformers[6] от HuggingFace. Модели, обученные в данной работе, также доступны в этом каталоге и в библиотеке DeepPavlov[7]

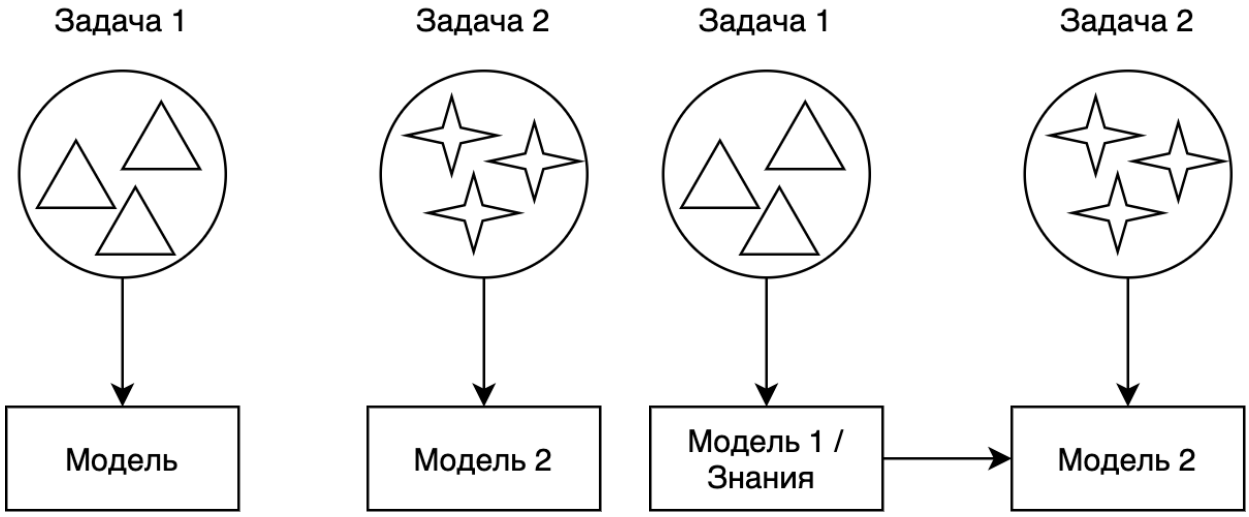
2.3 Перенос знаний с обученных языковых моделей BERT

Модели на основе BERT позволили улучшить качество решения многих задач обработки естественного языка для английского языка. В рамках диссертационной работы впервые была обучена модель BERT для русского языка.

[4] https://github.com/google-research/bert/blob/master/multilingual.md
[5] https://huggingface.co/models
[6] https://github.com/huggingface/transformers
[7] http://docs.deeppavlov.ai/en/master/features/pretrained_vectors.html#bert

Обучение языковых моделей типа BERT требует большого объема вычислительных ресурсов и длительного по времени обучения (4 дня вычислений на 4 Cloud TPUv2, что эквивалентно месяцу вычислений на DGX-1 с 8 графическими ускорителями NVIDIA Tesla P-100 16 Гб). Для того, чтобы ускорить процесс обучения предлагается использовать методы переноса знаний (transfer learning) из уже обученных языковых моделей BERT.

В общем случае, под переносом знаний подразумевается процесс обучения новой модели на новой задаче с переиспользованием знаний модели, обученной ранее на схожей задаче или схожем домене (рисунок 2.6). Например, знание полученное при обучении предсказания следующего (или скрытого) слова многоязычной языковой модели может быть переиспользовано для обучения одноязычной языковой модели (для языка на котором многоязычная модель была уже обучена ранее). Такой перенос знаний на практике может быть реализован следующим образом: в качестве начальной инициализации весов новой модели берутся веса модели, обученной ранее на схожей задаче, и производится обучение на новой задаче.



а) Независимое обучение двух моделей на двух задачах.

б) Перенос знаний. Обучение модели на задаче 2 с использованием знаний полученных моделью 1 на задаче 1.

Рисунок 2.6 — Независимое обучение двух моделей и обучение с переносом знаний (transfer learning).

Как было описано в разделе 1.1 нейросетевые языковые модели состоят не только из значений параметров нейросети, но еще и из словаря. Словарь фиксируется до начала обучения и в процессе обучения не изменяется, поэтому

в явном виде перенос знаний к словарям не применим. Это является проблемой, если нам нужен словарь отличный от того, что используется в модели с которой осуществляется перенос знаний. Есть несколько способов, которыми можно определить словарь для новой модели:

1. Использовать словарь модели с которой осуществляется перенос знаний;
2. Использовать пересечение словаря модели с которой осуществляется перенос знаний и нового словаря;
3. Использовать новый словарь.

Первый способ может быть использован в случаях если для новой задачи доступен только малый объем обучающих данных (словарь построенный на небольшом объеме данных может быть некачественным и сильно смещенным), или в данных представлен такой же домен и языки (в этом случае нет необходимости в другом словаре).

Второй способ позволяет отказаться от хранения неиспользуемых сабтокенов, как в словаре, так и в матрице с векторными представлениями сабтокенов. Однако, при этом новый словарь может оказаться небольшого размера, что приведёт к разделению слов на более мелкие сегменты, и, следовательно, увеличению длин последовательностей.

В силу того, что блоки `self-attention` в архитектуре Трансформер имеют квадратичную от длины вычислительную сложность (раздел 2.1.2), удлинение входных последовательностей увеличивает время работы модели. Эту проблему может отчасти решить третьим способом — уменьшить длину последовательностей за счет более подходящего словаря, но использование нового словаря требует внесения изменений в матрицу векторных представлений. Для новых сабтокенов из нового словаря нет векторных представлений в матрице векторных представлений, а значит их нужно добавить. Наивным образом векторные представления новых сабтокенов можно инициализировать случайными векторами.

Более эффективно, что показывается в разделе 2.3.5, инициализировать усредненным вектором суммы векторных представлений сабтокенов, полученных при разбиении нового сабтокена на сабтокены из словаря модели с которой осуществляется перенос знаний.

2.3.1 Инициализация векторных представлений для новых сабтокенов

При переносе знаний на модель с новым словарем сабтокенов возникает проблема несоответствия словарей и следовательно матрицы векторных представлений сабтокенов. Пусть V^{src} — словарь модели с которой осуществляется перенос знаний, а V^{tgt} — словарь модели на которую переносятся знания. E^{src}, E^{tgt} - матрицы векторных представлений, E_i^{src} — векторное представление сабтокена V_i^{src}. Матрица E^{tgt} неизвестна, она будет собираться из матрицы E^{src}. Мы также полагаем, что оба словаря V^{src} и V^{tgt} — это словари сабтокенов полученные алгоритмом BPE (Byte-Pair Encoding) и включают в себя сабтокены длины 1 (т. е., символы).

Для того, чтобы собрать матрицу E^{tgt} для сабтокенов $V_i^{tgt} \in V^{src}$ присваиваем $E_i^{tgt} = E_i^{src}$, но еще остаются V_i^{tgt} не содержащиеся в V^{src}. В таком случае, для тех $V_i^{tgt} \notin V^{src}$, нужно получить вектора E_i^{tgt}:

1. Наивный способ: случайная инициализация $E_i^{tgt} \sim \mathcal{N}(\mu, \sigma)$, где μ и σ — это среднее и дисперсия значений из матрицы векторных представлений E^{src}.
2. Усреднение векторных представлений сабтокенов из E^{src}.

Опишем подробнее второй способ. Разобьем каждый сабтокен V_i^{tgt}, отсутствующий в словаре V^{src}, алгоритмом BPE на сабтокены из словаря V^{src} и получим разбиение на сабтокены $T_{V^{src}}(V_i^{tgt}) = t_{i,1}, t_{i,2}, \ldots, t_{i,N}$. Тогда векторное представление сабтокена V_i^{tgt} определяется как среднее векторных представлений составляющих его сабтокенов словаря V^{src}:

$$E_i^{tgt} = \frac{\sum_{k=1}^{N} e(t_{i,k})}{N}, \tag{2.2}$$

где e - функция получения векторного представления сабтокена из матрицы E^{src}. Например, слово «трансформер» отсутствует в словаре V^{src}. Результат разбиения на сабтокены с помощью BPE токенизация $T_{V^{src}}(\textit{трансформер})$ производит два сабтокена «*трансформ*» и «*\#\#ер*». Тогда инициализацией слова «*трансформер*» будет среднее векторных представлений сабтокенов «*трансформ*» и «*\#\#ер*». Данный подход с усреднением идейно близок к построению векторных представлений слов из составляющих его N-грамм [4].

2.3.2 Перенос знаний с многоязычных на языко-специфичные языковые модели

В качестве многоязычной языковой модели будем рассматривать многоязычный BERT (BERT-base, Multilingual Cased[8]). Многоязычный BERT обучен на сегментах Википедии для 104 языков и размер его словаря составляет примерно 120 тысяч сабтокенов. В эти 104 языка входит и русский, но как было показано по результатам тестирования модели [9] языко-специфичные модели для английского и китайского языков превосходят по качеству многоязычную модель BERT на задаче NLI. Поэтому возникает мотивация обучить языко-специфичные модели и для других языков (русский) и для более узких групп языков (славянские), используя уже обученную многоязыковую модель BERT.

Выбор параметров многоязычного BERT при переносе на русский язык

Не все параметры многоязычного BERT могут быть использованы для русского языка. Так, словарь многоязычного BERT состоит из примерно 120 тысяч сабтокенов из которых только 25 тысяч могут быть отнесены к русскому языку (содержат кириллические символы, цифры, пунктуацию), что составляет около 20%. В многоязычном BERT 180 миллионов параметров из которых половину занимает матрица векторных представлений сабтокенов. Оценим сверху, какая доля параметров многоязычного BERT может быть использована для русского языка: $50\% + 50\% \cdot 20\% = 60\%$. Остальные 40% никак не задействуются для русского языка, а 50% из этих 60% используются одновременно всеми 104 языками. Поэтому можно либо уменьшить число параметров в модели для русского языка, либо сохранить число используемых параметров за счет построения словаря для русского языка. Аргументом за сохранение числа параметров и построение нового словаря является то, что новый словарь позволит уменьшить длины

[8] https://github.com/google-research/bert/blob/master/multilingual.md
[9] https://github.com/google-research/bert/blob/master/multilingual.md

входных последовательностей, так как меньшая доля слов будет разбиваться на сабтокены.

2.3.3 Перенос знаний языковых моделей с одного домена на другой

Аналогично переносу знаний для обучения языко-специфичных языковых моделей, можно обучить языковые модели под определенный домен. В качестве домена, для которого будем адаптировать универсальные языковые модели, рассматриваем «разговорные» данные: социальные медиа (d3.ru, Pikabu), социальные сети (Twitter, Facebook) и диалоги (opensubtitles, DailyDialogs). В результате, переноса знаний будут получены разговорные языковые модели `BERT` как для русского, так и для английского языка.

2.3.4 Данные для обучения языковых моделей на базе архитектуры Трансформер

RuBERT Модель `RuBERT` для русского языка была обучена на наборе текстов из русского сегмента Википедии и новостных статьях с сайта Lenta.ru[10]. Общий объем текстовых данных после фильтрации и предобработки составил 6,5 Гб, из них около 80% данных из Википедии.

Разговорный RuBERT Модель `Разговорный RuBERT` для русского языка была обучена на текстах из следующих источников: Социальные сети из корпуса Taiga[11] [81], русскоязычные субтитры из OpenSubtitles [82], комментарии с сайтов Pikabu и Dirty. Общий объем текстов 3,3 Гб.

Разговорный BERT Модель `Разговорный BERT` для английского языка была обучена на текстах из следующих источников: англоязычные сообщения из Twitter и комментарии из Reddit, диалоги из DialyDialogs [83] и

[10] https://github.com/yutkin/Lenta.Ru-News-Dataset
[11] https://tatianashavrina.github.io/taiga_site/

OpenSubtitles [82], Debates [84], Blogs [85], комментарии из Facebook[12]. Общий объем текстов 21,7 Гб.

Славянский BERT (Bg, Cz, Pl, Ru) Модель `Славянский BERT` (Bg, Cz, Pl, Ru) была обучена на следующих источниках: Wikipedia на болгарском, чешском, польском и русском языках. Для каждого языка данные Wikipedia были сбалансированы таким образом, чтобы пропорция языков была равной. Так же были добавлены русские новостные данные Lenta.ru[10]. Общий объем текстов 2,4 Гб.

2.3.5 Обучение языковых моделей с архитектурой Трансформер

Обучение языковых моделей с архитектурой Трансформер производится на наборах данных, представленных в разделе 2.3.4.

Для моделей `RuBERT` и Славянский `BERT` в качестве инициализации был использован многоязычный `BERT` (`BERT-base, Multilingual Cased`[8]). Веса разговорного `RuBERT` инициализировались весами модели `RuBERT`, веса разговорного `BERT` инициализировались весами модели `BERT`. Далее будет описан процесс пересборки словаря для перечисленных моделей и даны детали обучения.

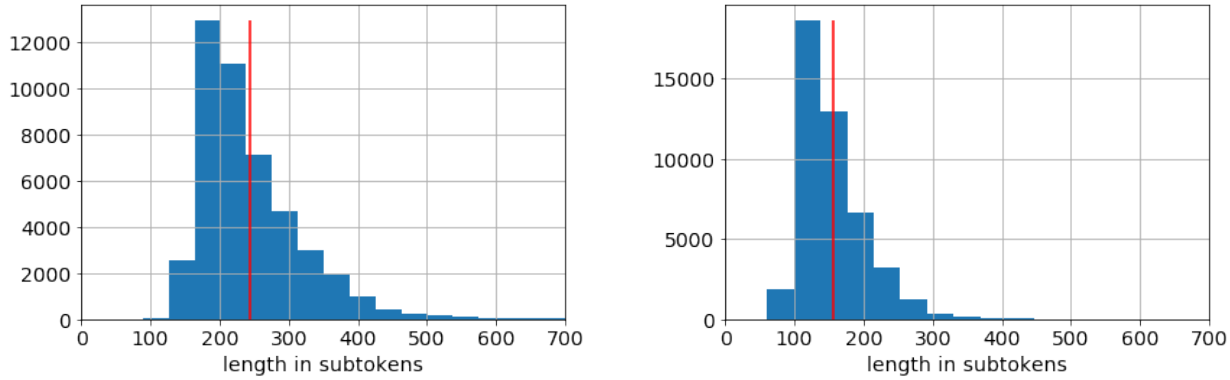
Пересборка словаря

Для данных моделей были пересобраны BPE словари сабтокенов. Новый BPE словарь был построен (обучен) на основе тех же данных, на которых производилось дообучение (данные описаны в разделе 2.3.4). Построение нового словаря позволяет значительно снизить среднюю длину слова в сабтокенах. Снижение средней длины слова приводит к уменьшению средней длины предложений, что уменьшает вычислительные затраты. Снижение длины особенно важно в силу ограниченности количества позиционных векторов, а так же

[12] https://github.com/jbencina/facebook-news

квадратичной вычислительной сложности архитектуры Трансформер в зависимости от длины входной последовательности. На рисунке 2.7 приведены примеры распределений длин входных последовательностей сабтокенов для многоязычной модели и модели `RuBERT` на текстах из набора данных SDSJ Задача B[13] (SberQuAD [86]).

Пересборка словаря позволила уменьшить среднюю длину последовательностей примерно в 1,55 раз с 243,28 (многоязычный словарь) до 156,18 (пересобранный словарь) на этом наборе данных (ускорение вычислений прямого прохода для модели размера `BERT-base` примерно в 1,4, рисунок 2.8).



а) `Многоязычный BERT` б) `RuBERT`

Рисунок 2.7 — Распределение длин текстов в сабтокенах (данные SDSJ Задача B). Вертикальной красной линией обозначено среднее значение. Средняя длина разбиения текста на сабтокены уменьшилась примерно в 1,55 раз после пересборки словаря.

Влияние инициализации на динамику обучения

Для оценки влияния инициализации на динамику обучения были произведены эксперименты с тремя различными инициализациями:

– случайная инициализация параметров;
– инициализация с помощью модели `многоязычный BERT` (векторные представления новых сабтокенов инициализированы случайно);
– инициализация с помощью модели `многоязычный BERT` и пересборка векторных представлений сабтокенов, описанная в разделе 2.3.1.

[13] https://sdsj.sberbank.ai/static/2017/en/contest.html

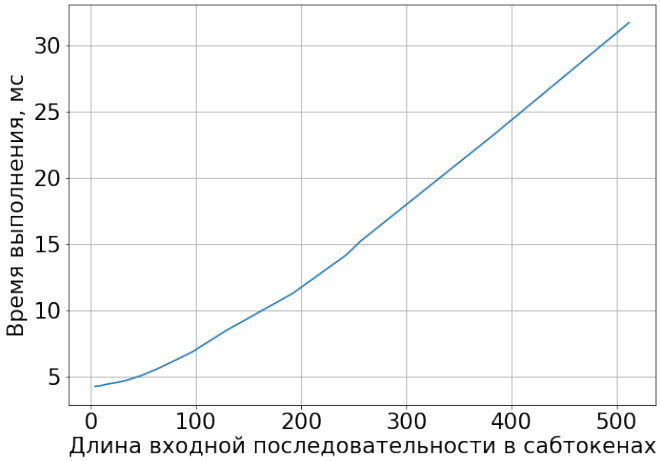


Рисунок 2.8 — Время вычислений прямого прохода для модели размера `BERT-Base` (12 слоев, 12 голов внимания, 768 размер скрытого состояния) в миллисекундах для входных последовательностей разных длин.

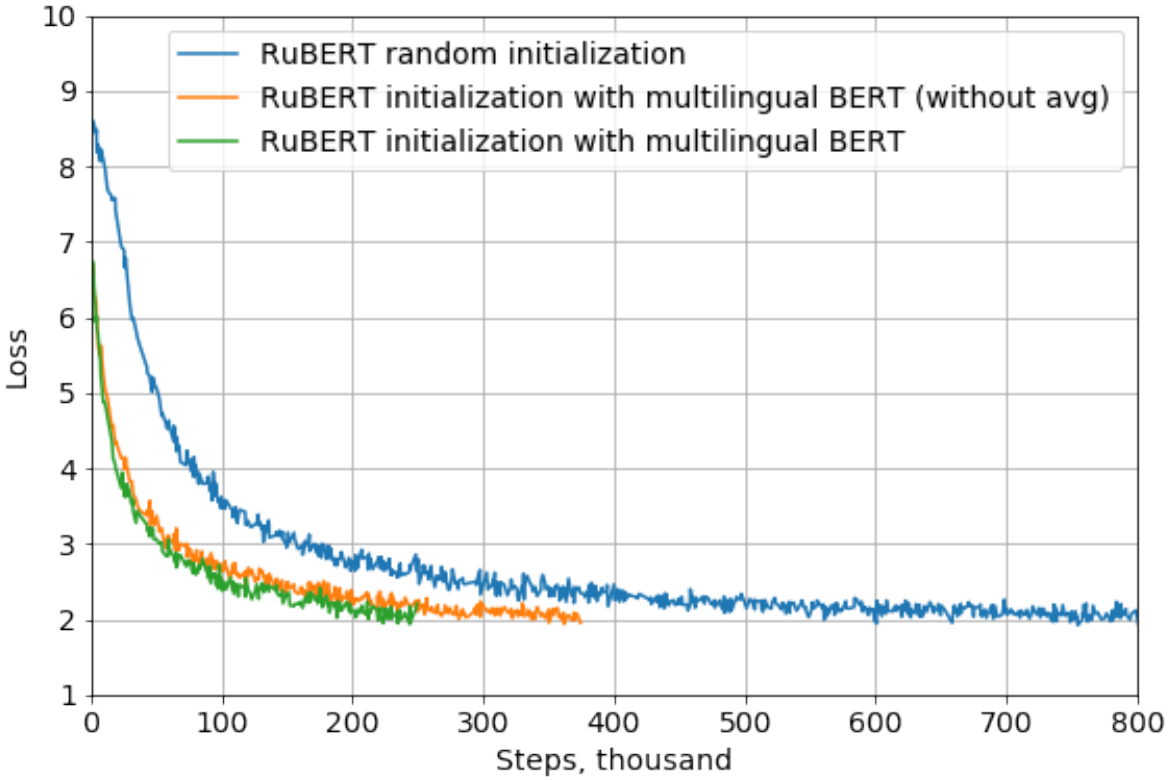


Рисунок 2.9 — Динамика обучения языковой модели для русского языка до достижения одинакового значения функции потерь (loss function).

На рисунке 2.9 изображены кривые обучения для трёх различных инициализаций. Обучение производилось вплоть до момента достижения функцией потерь значения 2.0. Обучение моделей на 120 тысяч шагов занимает около 24 часов на одном DGX-1 с 8 графическими ускорителями NVIDIA Tesla P-100 16 Гб. Как можно видеть, инициализация `многоязычным BERT` позволяет значи-

тельно сократить время обучения даже при условии случайной инициализации векторных представлений слов (с семи до трех дней). В то же время, инициализация векторных представлений с помощью усреднения элементов новых слов в словаре позволяет достичь еще большей скорости сходимости (примерно два дня вычислений).

Детали процесса обучения

При обучении языковых моделей были использованы следующие гиперпараметры:

- Шаг обучения (learning rate): $2 \cdot 10^{-5}$
- Оптимизатор: Adam with Weight Decay
- L2 регуляризация: 10^{-2}
- Длины входных последовательностей: 128 и 512
- Размеры батчей: 256 и 72

В силу того, что длина последовательности ограничивает максимальный размер батча для каждой из моделей тренировка производилась в два этапа: с короткой длиной последовательности (128 сабтокенов) и большим размером батча (256) и большей длиной (512 сабтокенов), но небольшим размером батча (72). Как правило больший размер батча обеспечивает более стабильное обучение Трансформеров с более высокими результатами [87]. Однако, при обучении на коротких последовательностях обновляется лишь часть позиционных векторных представлений, отвечающих за позиции с 1 по 128. Поэтому, сначала ведётся обучение на коротких последовательностях, а затем уже производится дообучение на длинных. В таблице 3 приведены количества шагов обучения для каждой из описанных моделей при различных длинах входных последовательностей.

Обучение языковых моделей на 120 тысяч шагов занимает около 24 часов на одном DGX-1 с 8 графическими ускорителями NVIDIA Tesla P-100 16 Гб. Для обучения языковых моделей был модифицирован оригинальный код[14] обучения моделей `BERT` для поддержки обучения на нескольких графических

[14] https://github.com/google-research/bert

ускорителях и выложен в открытый доступ[15]. Оригинальный код поддерживает обучение только на одном графическом ускорителе либо на тензорных процессорах (TPU[16]).

Модель	Шагов обучения (128)	Шагов обучения (512)
RuBERT	4M	300k
Разговорный RuBERT	4M	2M
Разговорный BERT	6M	2M
Славянский BERT	1M	1M

Таблица 3 — Количество шагов обучения для различных моделей BERT при обучении на разных длинах входных последовательностей (приведены приближённые значения).

В главе 2 была описана архитектура Трансформер, методика предобучения языковых моделей BERT. Основные результаты и выводы:

- Предложен метод переноса знаний с обученных языковых моделей BERT для их дообучения;
- Показано, что предложенный метод переноса знаний позволяет ускорить процесс предобучения языковых моделей;
- Частью предложенного метода переноса знаний является пересборка словаря под новый язык или домен. Пересобранные словари позволяют уменьшить длины входных последовательностей, а значит ускорить работу моделей и уменьшить требования к доступной оперативной или видеопамяти.
- Предложенный метод переноса знаний применен для переноса знаний с многоязычного BERT на языко-специфичные RuBERT для русского языка и Славянский BERT для болгарского, чешского, польского и русского языков;
- Предложенный метод переноса знаний применен для переноса знаний с моделей общего домена BERT и RuBERT на разговорный домен для английского и русского языков;

[15] https://github.com/deepmipt/bert/tree/feat/multi_gpu
[16] https://cloud.google.com/tpu

– Все предобученные языковые модели выложены в открытый доступ в библиотеке DeepPavlov[17];
– Предложенный метод переноса знаний опубликован в работе «Adaptation of Deep Bidirectional Multilingual Transformers for Russian Language» [19].

Применение обученных языковых моделей к различным задачам обработки естественного языка и анализ результатов представлен в главах 3, 4, 5.

[17] http://docs.deeppavlov.ai/en/master/features/pretrained_vectors.html#bert

Глава 3. Применение языковых моделей на базе архитектуры трансформер к задачам обработки естественного языка

Для оценки качества языковых моделей BERT описанных в главе 2 проверим их на некоторых задачах обработки естественного языка:

- классификация текстов;
- разметка последовательности; (sequence tagging / labeling)
- разрешение кореференции;
- вопросно-ответные системы (question-answering systems и machine reading comprehension).

Первые две задачи будут описаны и разобраны в данной главе, задача разрешения кореференции в главе 4 и вопросно-ответные системы в главе 5.

3.1 Классификация текстов

Большое число задач, связанных с текстом, сводится к задаче классификации: детекция спама, определение эмоциональной окраски, намерений пользователей диалоговой системы, выявление оскорблений и неуместной речи, определение авторства и тематики текста.

Предобученные языковые модели могут использоваться как источник признаков для моделей классификации, т. е. по ходу обучения классификатора параметры языковой модели не обновляются. Также все параметры предобученной языковые модели можно дообучать на целевой задаче классификации. Первый способ менее затратен по вычислительным ресурсам, а второй позволяет достигать лучших результатов [68].

3.1.1 Описание подхода к классификации с использованием языковых моделей на базе архитектуры Трансформер

Для решения задачи классификации текстов с использованием языковых моделей BERT добавляется один полносвязный линейный слой с размерностями[1] $768 \times num\_classes$, где 768 — размерность выходного слоя сети, $num\_classes$ — число классов. В этот линейный слой подается выход с последнего слоя сети для специального токена $[CLS]$, к которому применили полносвязный слой (с размерностями 768×768) с функцией активации гиперболический тангенс:

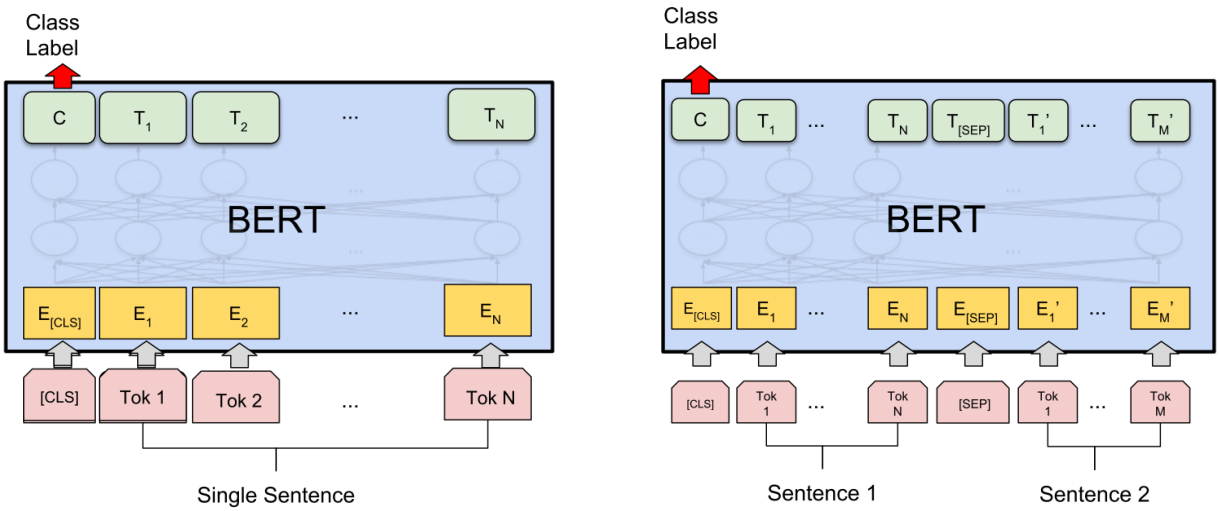
$$P(y|x) = softmax(FC_2(tanh(FC_1(BERT^N_{[CLS]}(x))))), \tag{3.1}$$

где $P(y|x)$ — распределение вероятностей по классам, $BERT^N_{[CLS]}(x)$ — выход с последнего слоя N языковой модели для токена $[CLS]$ и входных данных x, FC_1 — полносвязный линейный слой 768×768, FC_2 — полносвязный линейный слой $768 \times num\_classes$. Полносвязные слои FC_1 и FC_2 обучаются на размеченных данных соответствующей задачи классификации. Слой FC_1 уже предобучен на задаче предсказания следующего предложения.

Для классификации одного или двух предложений (текстов) используется одна архитектура, отличается только способ подачи данных в модель BERT:

- **Одно предложение (текст).** На вход сети подается текст разбитый на токены, перед началом текста добавляется специальный токен $[CLS]$. Под классификацию одного текста подходят, например, определение эмоциональной окраски и оскорблений.
- **Пара предложений (текстов).** Оба текста разбиваются на токены. Тексты разделяются специальным токеном $[SEP]$ и перед началом первого текста добавляется токен $[CLS]$. Применяется, например, для определение парафраз.

[1] Размерность скрытых слоев 768 указана для моделей размера BERT-Base, для моделей BERT-Large — 1024.



а) Классификация одного предложения (текста).

б) Классификация двух предложений (двух текстов).

Рисунок 3.1 — Применение языковой модели на базе архитектуры Трансформер для задачи классификации. Рисунок взят из оригинальной статьи BERT [8].

3.1.2 Описание данных

ParaPhraser [88] — набор данных на русском языке для идентификации парафраз. Два предложения являются парафразами, если они обладают одинаковым значением, но, при этом, могут быть по-разному сформулированы. Этот набор данных состоит из 7227/1924 тренировочных/тестовых пар предложений, которые помечены как точные парафразы (precise paraphrases), почти парафразы (near paraphrases) или непарафразы (non-paraphrases). Одним из возможных подходов к идентификации парафраз является бинарная классификация: первый класс точный и почти парафраз, второй класс - непарафраз.

RuSentiment [89] — набор данных на русском языке для определения сентимента (эмоциональной окраски) постов ВКонтакте (VK), самой популярной социальной сети в России. Будучи выложенным в 2018 году, он стал одним из крупнейших русскоязычных наборов данных для анализа сентимента с 30,521 постом. Каждый пост помечен одним из пяти классов: нейтральный (neutral), негативный (negative), положительный (positive), речевой акт (speech act) и пропуск (skip). Неформальный язык, представленный в RuSentiment, услож-

няет задачу анализа сентимента для модели RuBERT, обученной на Википедии и новостных статьях.

Yelp Reviews — набор данных на английском языке для определение сентимента отзывов оставленных на сайте Yelp[2], классификация на 5 классов: число звезд поставленных в отзыве от 1 до 5.

Insults — набор данных на английском языке для определения оскорбительных комментариев в социальных сетях представленный в рамках соревнования на Kaggle[3]. Данные размечены на два класса: есть ли в комментарии оскорбление или нет. Разбиение на обучающие, валидационные и тестовые данные брались как в исходном соревновании.

Stanford Sentiment Treebank [90] — набор данных на английском языке для определения сентимента отзывов на фильмы[4]. Классификация на 5 классов: сильно положительные, положительные, нейтральные, негативные, сильно негативные.

3.2 Разметка последовательности

Задача разметки последовательности (sequence labeling) применительно к текстам сводится к задаче классификации для каждого элемента последовательности, т. е. слова. Примеры такой задачи: разметка именованных сущностей (NER), частей речи (POS), синтаксическая разметка, расстановка знаков препинания.

В этом разделе будет описано применение модели BERT к задаче разметки именованных сущностей, но идеи и подход применимы и к другим задачам.

Реализация модели для разметки именованных сущностей и эксперименты были проведены Архиповым Михаилом и Трофимовой Марией. Куратов Юрий обучил и подготовил модели Славянский BERT и RuBERT для дообучения на

[2] https://www.yelp.com/dataset/challenge
[3] https://www.kaggle.com/c/detecting-insults-in-social-commentary
[4] https://nlp.stanford.edu/sentiment/index.html

целевой задаче. Результаты по экспериментам на наборе данных BSNLP-2019 опубликованы в совместной статье [22].

3.2.1 Описание подхода к разметке последовательностей с использованием языковых моделей на базе архитектуры Трансформер

Модель `BERT` работает с текстом на уровне сабтокенов и на выходе из последнего слоя можно получить контекстно-зависимые векторные представления сабтокенов. Затем к векторным представлениям тех сабтокенов, которые являются началами слов, применяется линейный полносвязный слой (аналогично, тому как это делается в задаче классификации) для формирования предсказаний. Линейный слой формирует предсказания для каждого слова независимо, а это может приводить к нежелательным ошибкам в случае, если существует сильные зависимости между предсказаниями для разных слов. В таких случаях для задач разметки последовательностей применяют условные случайные поля (CRF, conditional random fields) [91]. Модель `BERT` с линейным и CRF слоем изображена на рисунке 3.2, также на этом рисунке показано как получить предсказания для слов, если модель работает на уровне сабтокенов.

3.2.2 Описание данных

Collection 3 [92] — набор данных для оценки качества алгоритмов автоматического извлечения именованных сущностей из текстов на русском языке. В качестве основы для разметки этих данных взята коллекция Persons-1000 [93], подготовленная Исследовательским центром Искусственного интеллекта Института программных систем РАН.

BSNLP-2019. В рамках минисимпозиума (workshop) Balto-Slavic Natural Language Processing 2019 [94] было проведено соревнование по извлечению именованных сущностей. Площадкой для этого мероприятия стала конференция

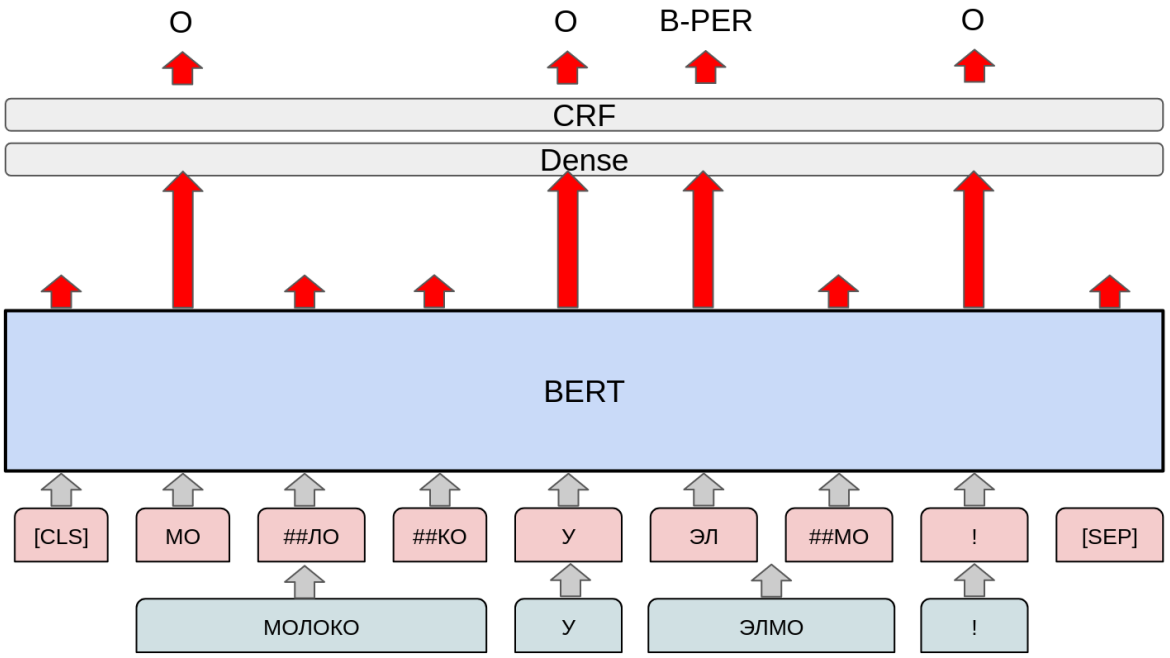


Рисунок 3.2 — Применение модели BERT для задачи разметки последовательности.

ACL-2019. В рамках соревнования был опубликован набор веб-документов на четырех славянских языках (болгарский, чешский, польский, русский) для задачи распознавания пяти типов именованных сущностей, а именно: персона (PER), локация (LOC), организация (ORG), событие (EVT) и продукт (PRO). Веб-документы поделены на отдельные сборники новостных текстов, где каждый сборник содержит новости о какой-то одной медиа-личности или событии. Описываемый набор данных был получен путем скачивания новостных веб-страниц в формате HTML и грамматического разбора в целях выделения текстов новостей.

Во время соревнования были доступны два размеченных сущностями сборника: сборник новостей про *Asia Bibi* и *Brexit*. Так как последний сборник больший по числу текстов, сборник про *Brexit* был использован в качестве обучающей выборки, а сборник про *Asia Bibi* в качестве валидационной.

3.2.3 Метрики качества

Span F-1 — F-1 мера на уровне упоминаний сущностей, т. е. фрагментов текста (spans). Упоминание сущности считается определенным правильно, если все токены из которых оно состоит размечены верно.

В рамках соревнования BSNLP были введены еще три метрики на уровне сущностей: RPM (Relaxed Partial Matching), REM (Relaxed Exact Matching), SM (Strict Matching), которые отличаются тем, как понимается, что именованная сущность определена верно:

- relaxed (не строгое) — сущность считается определена верно, если хотя бы для одного из её упоминаний в тексте модель сделала верное предсказание;
- strict (строгое) — сущность считается определена верно, если для всех её упоминаний в тексте модель сделала верное предсказание;
- partial (частичное) — упоминание сущности определено верно, если хотя бы один его токен предсказан верно;
- exact (точное) — упоминание сущности определено верно, если все его токены предсказаны верно.

3.3 Результаты на задачах классификации и разметки последовательностей

В таблицах 4, 5, 6 приведены результаты полученные на описанных в разделе 3.1.2 наборах данных.

Предложенная методика переноса знаний с многоязычного `BERT` на `RuBERT` и дальнейшее дообучение были протестированы на задачах определения парафраз (ParaPhraser) и эмоциональной окраски (RuSentiment) для русского языка. На обеих задачах модель `RuBERT` показывает лучшие результаты по сравнению с моделью многоязычный `BERT` и улучшает ранее опубликованные результаты. На задаче определения эмоциональной окраски была еще проверена модель Разговорный `RuBERT`, которая показывает еще более высокие значения метрик (76.3 F-1), чем RuBERT (72.63 F-1). Такая разница объясняется тем, что домен в наборе данных RuSentiment совпадает с доменом данных, на которых обучалась модель Разговорный RuBERT. На наборе данных ParaPhraser наблюдается тоже самое, модель `RuBERT` показывает результат лучше, чем Разговорный `RuBERT` из-за домена данных.

Также эффективность обучения языковой модели на данных того же домена, что и домен данных целевой задачи, была продемонстрирована для

Таблица 4 — ParaPhraser. Сравниваются метрики полученные с использованием предобученных языковых моделей на базе архитектуры Трансформер с моделями из работ других авторов. Все результаты приведены для non-standard режима (для обучения можно использовать любые дополнительные данные) [88]. Результаты для `BERT` моделей получены усреднением результатов после 5 запусков обучения.

Модель	F-1	Accuracy
Нейросетевой классификатор [88]	79.82	76.65
Классификатор и лингвистические признаки [88]	81.10	77.39
Машинный перевод и семантическая близость [95]	78.51	81.41
Многоязычный BERT	85.48 ± 0.19	81.66 ± 0.38
Разговорный RuBERT	86.54 ± 0.62	83.36 ± 0.83
RuBERT	$\mathbf{87.73} \pm 0.26$	$\mathbf{84.99} \pm 0.35$

Таблица 5 — RuSentiment. Использовалось только случайно выбранное подмножество данных для обучения (21,268 примеров, разбиение от авторов набора данных). Результаты для `BERT` моделей получены усреднением результатов после 5 запусков обучения.

Модель	F-1	Precision	Recall
Логистическая регрессия [89]	68.84	69.53	69.46
Метод опорных векторов (Linear SVC) [89]	68.56	69.46	69.25
Градиентный бустинг [89]	68.48	69.63	69.19
Нейросетевой классификатор [89]	71.64	71.99	72.15
Многоязычный BERT	70.82 ± 0.75	-	-
RuBERT	72.63 ± 0.55	-	-
Разговорный RuBERT	$\mathbf{76.22} \pm 0.75$	-	-

английского языка на примере трех наборов данных: Insults, Yelp Reviews, Stanford Sentiment Treebank (SST). Результаты для задачи классификации на этих трех наборах данных приведены в Таблице 6. Модель Разговорный `BERT` позволяет улучшить значения метрик на 1-2 пункта за счет более подходящего домена. Более значительное улучшение на примерно 4 пункта F-1 наблюдается для русского языка на данных RuSentiment. На наборе данных ParaPhraser (заголовки новостных статей) лучше себя показывает `RuBERT`, обученный на Википедии и новостных данных.

Таблица 6 — Сравнение модели BERT/RuBERT и Разговорного BERT/RuBERT на пяти задачах классификации. Домен первых четырех наборов данных больше близок к домену данных на которых обучался Разговорный BERT/RuBERT.

Набор данных [язык]	Метрика	BERT	Разговорный BERT
Insults [En]	ROC-AUC	86.1	**88.4**
Yelp Reviews [En]	Accuracy	67.8	**68.6**
SST [En]	Accuracy	64.4	**66.1**
RuSentiment [Ru]	F-1	72.63 ± 0.55	**76.22** ± 0.75
ParaPhraser [Ru]	F-1	**87.73** ± 0.26	86.54 ± 0.62

Разметка последовательностей с помощью моделей BERT проверялась на задаче распознавания именованных сущностей на наборах данных Collection3 для русского языка и набора данных BSNLP-2019 для болгарского, чешского, польского и русского языков. В таблице 7 приведены результаты для многоязычного BERT и RuBERT для русского языка.

Таблица 7 — Collection 3. Представлена метрика Span F_1 на тестовой выборке. Если модель доступна в библиотеке DeepPavlov, то рядом с названием модели стоит символ *.

Модель	Span F_1
Bi-LSTM и CRF * [91]	95.14
Многоязычный BERT и CRF	97.24
RuBERT и CRF *	97.79

В таблице 8 приведены результаты для многоязычного BERT и Славянского BERT на наборе данных BSNLP-2019. Как и для других задач и наборов данных видно, что модели адаптированные под нужный язык и домен превосходят по качеству более общую многоязычную модель BERT [22]. По результатам соревнования BSNLP-2019 модель Славянский BERT и CRF заняла первое место по метрикам REM, SM и второе место по метрике RPM [94].

Таблица 8 — BSNLP. Представлены метрики Span F_1 и RPM, REM, SM. Метрики на тестовой выборке, известные для последней модели, указаны в скобках.

Модель	Span F_1	RPM	REM	SM
Bi-LSTM и CRF [91]	75.8	73.9	72.1	72.3
Многоязычный BERT	79.6	77.8	76.1	77.2
Многоязычный BERT и CRF	81.4	80.9	79.2	79.6
Славянский BERT	83.5	83.8	82.0	82.2
Славянский BERT и CRF	**87.9**	**85.7** (90.9)	**84.3** (86.4)	**84.1** (85.7)

Следующие выводы можно сделать по результатам полученным в этой главе.

– Многоязычные языковые модели уступают в качестве моделям, подготовленным специально для языков, используемых в задаче.
– Языковые модели, обученные на данных из домена, соответствующего целевой задаче, показывают лучшие результаты, чем «универсальные» модели (`BERT`, `многоязычный BERT`, `RuBERT`).
– Обученные в рамках работы над диссертацией языко- и доменно- специфичные модели позволили улучшить ранее опубликованные результаты для разных задач обработки естественного языка (классификации и распознавания именованных сущностей) и для разных наборов данных (RuSentiment, ParaPhraser, Collection-3, BSNLP-2019).
– Предобученный `Славянский BERT` позволил получить модели, занявшие первое место по двум из трех метрик на соревновании BSNLP-2019 [94].
– Результаты, полученные в этой главе, опубликованы в работах «Adaptation of Deep Bidirectional Multilingual Transformers for Russian Language» [19] и «Tuning Multilingual Transformers for Language-Specific Named Entity Recognition» [22].

Глава 4. Разрешение кореференции и языковые модели

Задача разрешения кореференции заключается в следующем: в данном тексте, необходимо найти все упоминания (mentions) и указать, какие из них обозначают один и тот же объект действительности. Упоминания могут состоять из нескольких слов. Упоминаниями могут быть выражения в тексте, обозначающие некоторую сущность, явление (объекты действительности). Некоторые объекты могут быть представлены в тексте только одним упоминанием, такие упоминания называются синглтонами.

Задача разрешения кореференции может быть рассмотрена как задача кластеризации упоминаний, где кластера — цепочки кореферентов (coreference chains — все упоминания, ссылающиеся на один объект) в тексте. Например, анафора — частный случай кореференции, в котором значение местоимения можно восстановить по некоторому ранее упомянутому выражению (антецеденту), а катафору — по упоминанию находящемуся позже в тексте (постцеденту).

Рассмотрим отрывок из произведения Тургенева И. С. «Отцы и дети»:

> [Николай Петрович]$_1$ познакомился с [Фенечкой]$_2$ следующим образом. Однажды, года три тому назад, [ему]$_1$ пришлось ночевать на [постоялом дворе]$_3$ в отдаленном уездном городе. [Его]$_1$ приятно поразила чистота отведенной [ему]$_1$ комнаты, свежесть постельного белья. «Уж не немка ли [здесь]$_3$ [хозяйка]$_2$?» — пришло [ему]$_1$ на мысль; но [хозяйкой]$_2$ оказалась русская, [женщина лет пятидесяти]$_2$, опрятно одетая, с благообразным умным лицом и степенною речью. [Он]$_1$ разговорился с [ней]$_2$ за чаем; очень [она]$_2$ [ему]$_1$ понравилась.

В этом отрывке квадратными скобками выделены упоминания, а одинаковыми индексами отмечены упоминания, принадлежащие одной цепочке кореферентов. Можно увидеть примеры как анафоры: *Николай Петрович — он*, так и кореференции для разрешения которой не достаточно только текста, но и необходимы дополнительные знания о мире и логика: *Фенечка — хозяйка – женщина лет пятидесяти.*

Задача разрешения кореференции сложна для компьютерных систем, так как для ее решения нужно обладать не только умением извлекать и обрабатывать информацию представленную в явном и не явном виде в тексте, но и иметь понимание об устройстве мира, обладать здравым смыслом (англ. common sense — общее знание, которым обладают бóльшинство людей), учитывать актуальность знаний (Встретились $[\text{Обама}]_1$ и $[\text{Трамп}]_2$. $[\text{Президент}]_?$ торжественно поприветствовал первым). Из-за такой сложности задача разрешения кореференции была предложена, как альтернатива тесту Тьюринга Гектором Левеском [96], и был организован The Winograd Schema Challenge. Наиболее известная схема Винограда за авторством Левеска:

> 1. **Кубок** не влезает в коричневый **чемодан**, потому что **он** слишком большой.
> 2. **Кубок** не влезает в коричневый **чемодан**, потому что **он** слишком маленький.
>
> вопрос к каждому из примеров:
> **Он** обозначает **чемодан** или **кубок**?[1]

в ней, чтобы ответить на вопрос система должна понимать, что для того, чтобы объект можно было положить внутрь чего-либо, он должен быть меньше по размеру, чем тот, в который его кладут.

Еще один пример необходимости уметь разрешать кореференцию для машинного перевода с английского языка на русский:

> Англ.: The recent **novel** became a bestseller. **It** is very popular.
> Рус.: Недавний **роман** стал бестселлером. **Он** очень популярен.

перевод слова *it* на русский язык зависит от рода слова **novel**. Для предоставления качественного перевода системам машинного перевода нужно уметь разрешать подобные неоднозначности.

Разрешение кореференции находит свое применение в широком спектре задач обработки естественного языка, таких как извлечение фактов и информации из текстов, машинный перевод, информационный поиск, диалоговые системы. Данная проблема вызывает интерес и тем, что остается плохо решенной на данный момент, и является направлением активных исследований.

[1] Приведены переводы примеров 3 и 4 из https://cs.nyu.edu/faculty/davise/papers/WinogradSchemas/WSCollection.xml

Системы, основанные на машинном обучении, зависят от объема и качества доступных размеченных данных. Также, для разных языков вопрос наличия данных решен в разной степени. Если для английского языка есть корпуса достаточно больших размеров, то для русского языка ситуация усугубляется наличием наборов данных на порядок меньших объемов.

На практике возможны следующие варианты постановки задачи:

1. **Упоминания выделены из текста (gold mentions).** Система разрешения кореференции может использовать сам текст, дополнительную разметку (морфологическую, синтаксическую) и выделенные упоминания в тексте (например, известны их границы).
2. **Текст без выделенных упоминаний (full).** Система разрешения кореференции может использовать только сам текст и, возможно, дополнительную разметку. Выделение упоминаний является подзадачей для модели.

Метрики Основными метриками, которые используются для сравнения и оценки качества систем разрешения кореференции являются: MUC, B^3, $CEAF_{mention}$, $CEAF_{entity}$ [97]. На соревновании CoNLL-2012 Shared Task [98] для определения системы победителя считалось среднее значение *F-1* (*CoNLL-F-1*) между MUC, B^3 и $CEAF_{entity}$.

MUC [99]: основывается на числе связей между упоминаниями, которые нужно вставить или удалить для того, чтобы получить кластеризацию, соответствующую разметке.

B³ [100]: сначала считается точность и полнота определения для каждого упоминания m в тексте. Итоговая точность и полнота получается усреднением для каждого m:

$$Precision(m) = \frac{|c_{predicted}(m) \cap c_{true}(m)|}{|c_{predicted}(m)|},$$

$$Recall(m) = \frac{|c_{predicted}(m) \cap c_{true}(m)|}{|c_{true}(m)|},$$

где $c_{predicted}(m)$ — предсказанный для упоминания m кластер, $c_{true}(m)$ — верный кластер для упоминания m.

CEAF [101]: мера качества системы разрешения кореференции, которая может подсчитываться как для упоминаний, так и для цепочек кореференции ($CEAF_{mention}$, $CEAF_{entity}$):

$$Precision = \frac{\sum_i \varphi(c^i_{true}, c^{\alpha(i)}_{predicted})}{\sum_i \varphi(c^i_{predicted}, c^i_{predicted})} \tag{4.1}$$

$$Recall = \frac{\sum_i \varphi(c^i_{true}, c^{\alpha(i)}_{predicted})}{\sum_i \varphi(c^i_{true}, c^i_{true})}, \tag{4.2}$$

где для $CEAF_{mention}$ в качестве функции схожести φ используется:

$$\varphi_3(c_{true}, c_{predicted}) = |c_{true} \cap c_{predicted}|,$$

а для $CEAF_{entity}$:

$$\varphi_4(c_{true}, c_{predicted}) = \frac{2|c_{true} \cap c_{predicted}|}{|c_{true}| + |c_{predicted}|}.$$

В числителе формул для точности (формула 4.1) и полноты (формула 4.2) находятся суммы величин схожести φ для каждого верного кластера c^i_{true} и наиболее схожего на него из предсказанных $c^{\alpha(i)}_{predicted}$, где функция $\alpha(i)$ находит наиболее схожий кластер согласно функции φ:

$$\alpha(i) = \operatorname*{argmax}_j \varphi(c^i_{true}, c^j_{predicted}).$$

4.1 Обзор данных и методов для разрешения кореференции

С появлением первых наборов данных для разрешения кореференции (MUC-6[2] (1995), MUC-7, ACE-02[3] - ACE-05) начали разрабатываться методы основанные на машинном обучении, тогда как до этого в основном применялись эвристические подходы основанные на правилах и экспертных знаниях [102]. Первые наборы данных в некотором роде определили то, в каком виде сейчас решается задача разрешения кореференции. Затем, в 2011 и в 2012 годах в рамках конференции CoNLL были проведены соревнования по разрешению кореференции на новых наборах данных CoNLL-2011 (английский язык) [103] и

[2] https://cs.nyu.edu/grishman/COtask21.book_1.html

[3] https://catalog.ldc.upenn.edu/LDC2003T11

CoNLL-2012 (английский, арабский, китайский языки) [98]. CoNLL-2012 стал стандартом для проверки качества систем разрешения кореференции. В ходе этого соревнования были разработаны системы основанные на правилах, на машинном обучении и гибридные походы сочетающие в себе оба предыдущих. Реализация гибридного подхода участником *fernandes* [104] показала наилучший усредненный результат на всех трех языках соревнования. Для того, чтобы построить кластер размера N достаточно установить $N-1$ связь между упоминаниями, т. е. построить дерево. В качестве алгоритма машинного обучения использовался голосующий персептрон (voted perceptron) [105] для определения предсказания наличия связи между двумя упоминаниями, и строилось минимальное остовное дерево алгоритмом Эдмондса.

Нейронные сети были впервые применены к задаче разрешения кореференции в 2015 году в работе Уайзмана [106]. Полносвязные слои с нелинейностью были использованы для обучения более эффективного представления входных признаков.

Для русского языка первые наборы данных RuCor[107] появились в 2014 году в рамках соревнования на конференции "Диалог 2014"[4]. На тот момент времени, результаты соревнования показали (таблица 9), что для русского языка эвристические походы являются более эффективными, чем подходы, основанные на машинном обучении [108]. Эти результаты идут в разрез с тем, что для английского языка в то время преобладали методы, основанные на машинном обучении. Основной причиной этого является на порядок меньший объем размеченных данных для русского языка.

	Тип алгоритма	*Precision*	*Recall*	F_1
1	Правила+онтологии	0.82	0.70	0.76
2	Правила	0.71	0.58	0.64
3	Правила	0.63	0.50	0.55
4	Логистическая регрессия+онтологии	0.54	0.51	0.53
5	SVM+онтологии	0.58	0.42	0.49
6	Решающие деревья	0.36	0.15	0.21

Таблица 9 — Результаты участников соревнования Dialogue Evaluation 2014 на наборе данных RuCor [107].

[4] http://www.dialog-21.ru

В 2019 году, также в рамках соревнования на конференции "Диалог 2019 было организовано новое соревнование на новом наборе данных AnCor [109]. Эксперименты, которые будут описаны далее в диссертационной работе, строятся в основном на этом наборе данных, и результаты были получены в рамках указанного соревнования.

В таблице 10 приводится сравнительная характеристика наборов данных RuCor [107], AnCor [109], CoNLL-2012 [98] для разрешения кореференции. Объем данных для русского языка на порядок уступает доступным размеченным данным для английского.

Таблица 10 — Наборы данных для разрешения кореференции. Число упоминаний и кореферентных цепочек посчитано для обучающих + валидационных + тестовых данных.

Набор данных [Язык]	Число упоминаний	Число цепочек
RuCor [Ru] [107]	16558	3638
AnCor [Ru] [109]	28961	5678
CoNLL-2012 [En] [98]	194480	44221

Методы основанные на эвристиках и правилах преимущественно опираются на результаты морфологического, синтаксического и семантического анализа текста [110; 111]. Классические методы машинного обучение, такие как логистическая регрессия, метод опорных векторов (SVM) [112], решающие деревья обучаются на признаках, построенных вручную и извлеченных системами анализа текстов [113]. Для русского языка можно отметить работу Сысоева [114] в которой кореференция разрешается в несколько этапов: определение главного слова в упоминании, уточнение границ упоминания и классификация пар упоминаний на наличие между ними кореферентной связи. В качестве модели машинного обучения использовались решающие деревья, обученные на вручную построенных признаках, результатах морфологического и синтаксического разбора, разметке именованных сущностей и семантических кластерах на основе векторных представлений слов word2vec. Для каждого из этапов были обучены отдельные модели.

Описанные выше методы, основанные на машинном обучении, реализуют идею, которая в английской литературе называется mention-pair — для каждой пары упоминаний независимо принимается решение есть ли между ними кореферентная связь. Впервые mention-pair подход был применен в работах [115;

116]. Недостатком этого подхода является его локальность — решение о наличии кореферентной связи принимается без учета других упоминаний в тексте, также может быть нарушена транзитивность.

В 2017 году вышла работа «End-to-End Neural Coreference Resolution» [117], в которой одна модель и извлекает кандидатов в упоминания и ранжирует кандидатов в антецеденты (mention-ranking подход). Обучение одновременно извлекать упоминания и ранжировать кандидатов позволяет улучшить общее качество за счет того, что не возникает накопления ошибки от нескольких этапов: извлечение упоминаний, поиск антецедентов. Также, модель использует только сами тексты и не требует признаков от каких либо систем анализа текста. Модель из этой работы, и следующей за ней Higher-order Coreference Resolution with Coarse-to-fine Inference [118] будет далее использована, как базовая для экспериментов, и описана в разделе 4.2.1.

4.2 Описание экспериментов

Все проведенные эксперименты строятся на основе базовой модели e2e-coref (работы «End-to-End Neural Coreference Resolution» [117] и «Higher-order Coreference Resolution with Coarse-to-fine Inference» [118]). Сначала была проверена воспроизводимость результатов на наборе данных CoNLL-2012 [98]. Затем модель была адаптирована для русского языка и проверено ее качество на наборе данных RuCor [107]. Затем был проведен ряд экспериментов для русского языка с языковыми моделями `ELMo` и `RuBERT`.

4.2.1 Базовая модель

Базовая модель, основанная на работах «End-to-End Neural Coreference Resolution» [117] и «Higher-order Coreference Resolution with Coarse-to-fine Inference» [118], далее по тексту может называться как базовая модель, baseline или e2e-coref. Модель e2e-coref была выбрана в качестве базовой, так как на тот момент времени она показывала наилучшие результаты для английского

языка на наборе данных CoNLL-2012 [98; 117; 118]. Также, она работает только на текстах, и не требует никакой дополнительной разметки, так как сама извлекает упоминания из текста.

В качестве входных признаков модель использует предобученные векторные представления слов. Кроме того, она позволяет использовать и дополнительные вектора-признаки, что в итоге может повысить качество работы за счёт дополнительной информации. Формировать эти дополнительные вектора-признаки можно под конкретную задачу или набор данных (например, разделение на собеседников в диалоге). Кроме того, после некоторых модификаций, данная модель может работать, в том числе, и с уже заранее извлечёнными упоминаниями (gold mentions).

На рисунке 4.1 изображена схема e2e-coref модели, состоящая из:

- входных векторных представлений слов (word embeddings);
- двунаправленной рекуррентной сети с LSTM [28] ячейками (Bidirectional LSTM) для построения представлений фрагментов текста (Span representations);
- вычисления вероятности того, что фрагмент текста является упоминанием (Mention score);
- вычисления вероятности для упоминания быть антецедентом для другого упоминания (Antecedent score).

Идейно схему работы модели можно разбить на три этапа:

- **Первый этап. Определение упоминаний** $s_m(i)$. Для того чтобы эффективно предсказывать к какой цепочке кореферентов принадлежит упоминание, важно знать контекст в котором располагается упоминание, и его непосредственную структуру (упоминания могут состоять из нескольких слов). Для учёта контекста используется двунаправленная рекуррентная сеть с LSTM-ячейками [28] (Bi-LSTM), на вход которой подаются предобученные векторные представления слов из текста x. В результате получается набор скрытых состояний двунаправленной рекуррентной сети с LSTM-ячейками x^*, в которых есть контекстная информация о словах. Далее модель строит представления для упоминаний, применяя к скрытым состояниям Bi-LSTM механизм внимания [37], взвешивающий каждое слово внутри упоминания. Максимальная длина упоминания L ограничена и может быть подобрана, например,

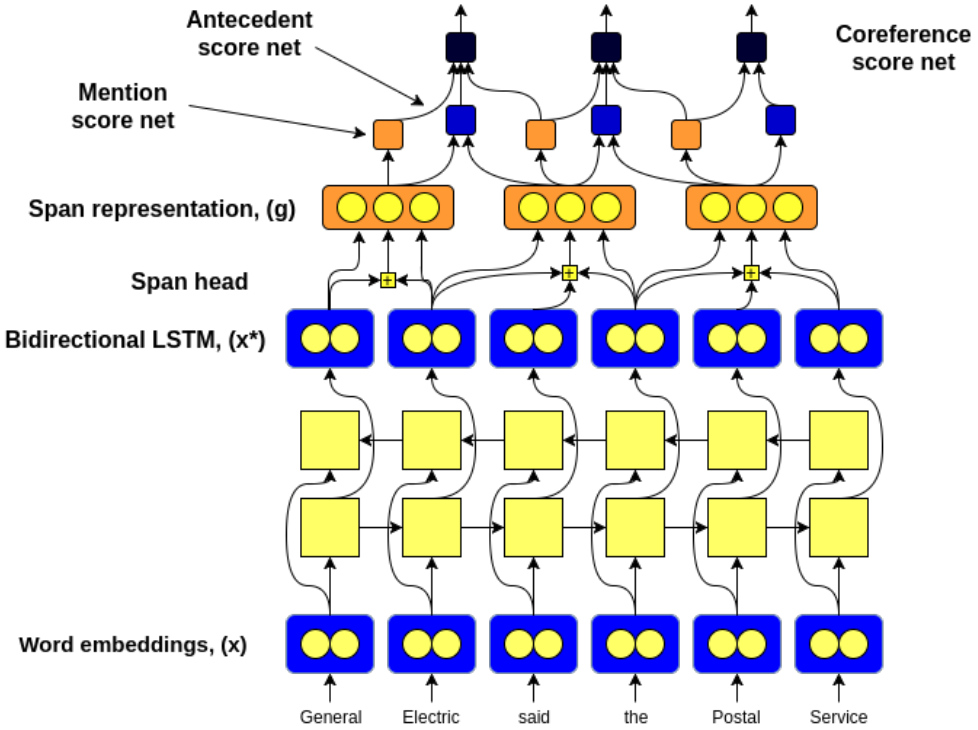


Рисунок 4.1 — Архитектура e2e-coref модели с извлечением упоминаний из текста (full pipeline).

на валидационном наборе данных.

$$\alpha_t = w_\alpha \cdot FC_\alpha(x_t^*),$$

$$a_{i,t} = \frac{\exp(\alpha_t)}{\sum_{k=start(i)}^{end(i)} \exp(\alpha_k)},$$

$$\hat{x_i} = \sum_{t=start(i)}^{end(i)} a_{i,t} \cdot x_t.$$

Итоговое векторное представление целого упоминания g_i получается в результате объединением векторных представлений первого слова в упоминании, последнего слова, взвешенного векторного представления всех слов в упоминании и дополнительного вектора-признака φ, если таковой имеется: $g_i = [x^*_{start(i)}, x^*_{end(i)}, \hat{x_i}, \varphi(i)]$. Полученный в результате объединения вектор подается на вход двухслойной полносвязной сети (Mention score на рисунке 4.1), которая обучается оценивать вероятность $s_m(i)$ того, что поданный на вход кандидат является упоминанием:

$$s_m(i) = w_m \cdot FC_m(g_i).$$

Для того, чтобы уменьшить объем вычислений для последующих этапов оставляется только доля λ от всех упоминаний с максимальной вероятность быть упоминанием.

– **Второй этап. Оценка для пары упоминаний** $s_a(i,j)$. Набор всех возможных пар полученных кандидатов упоминаний поступает на вход двухслойной полносвязной сети, которая обучается оценивать вероятность $s_a(i,j)$ того, что кандидаты, представленные в паре, являются антецедентом и анафором (Antecedent score net на рисунке 4.1):

$$s_a(i,j) = w_a \cdot FC_m([g_i, g_j, g_i \circ g_j, \varphi(i,j)]).$$

Также как и на предыдущем этапе, часть кандидатов в антецеденты отфильтровывается и берутся только максимум K из них для каждого из упоминаний.

– **Третий этап. Итоговая оценка принадлежности к одной цепочке кореферентов** $s(i,j)$. Полученные оценки для пары упоминаний, того что они принадлежат одной цепочке, и оценки того, что каждое из них является упоминанием суммируется и является финальной оценкой принадлежности к одной цепочке кореферентов:

$$s(i,j) = s_m(i) + s_m(j) + s_a(i,j),$$

и вероятность для антецедента y_i для упоминания i вычисляется при помощи функции softmax:

$$P(y_i) = \frac{e^{s(i,y_i)}}{\sum_{y' \in \mathcal{Y}(i)} e^{s(i,y')}}, \tag{4.3}$$

где $\mathcal{Y}(i)$ множество кандидатов в антецеденты для упоминания i.

Оценка $s(i,j)$ является локальной для пары упоминаний i и j. Для вычисления $s(i,j)$ используются представления упоминаний, построенные с помощью Bi-LSTM сети, которая обрабатывает предложения независимо. Эти представления не учитывают глобальную информацию из текста, и не учитывают другие упоминания. Проблема локальности оценки $s(i,j)$ не так сильно проявляется внутри одного предложения, так как Bi-LSTM сеть работает на уровне предложений. Для двух упоминаний из двух разных предложений глобальная информация о других упоминаниях в тексте отсутствует. Что затрудняет решение задачи.

Решение этой проблемы было предложено в работе Higher-order Coreference Resolution with Coarse-to-fine Inference [118].

Решение проблемы локальности модели e2e-coref. Для того, чтобы представления упоминаний g_i содержали в себе информацию о других упоминаниях во всем тексте было предложено итеративно обновлять представления g_i с помощью механизма внимания, применяя его ко всем антецедентам [118]. В качестве весов для механизма внимания на первой итерации используются вероятности рассчитанные по формуле 4.3 и векторные представления упоминаний $g_i^1 = g_i$. Представления упоминаний g_i^n для $n > 1$ с каждой итерацией обновляются средневзвешенным представлением антецедентов:

$$a_i^n = \sum_{g_j^n \in \mathcal{Y}(i)} P(g_j^n) \cdot g_j^n,$$

$$g_i^{n+1} = f_i^n \circ g_i^n + (1 - f_i^n) \circ a_i^n,$$

где f_i^n управляет тем, какую долю информации нужно сохранить от самого упоминания, а какую нужно записать от антецедентов:

$$f_i^n = \sigma(W[g_i^n, a_i^n]).$$

Число итераций $N = 2$ было подобрано на валидационном наборе данных CoNLL-2012[98] в работе авторов модели [118].

Модификация для работы с уже извлеченными упоминаниями. Эта же модель может быть использована при варианте постановки задачи, когда упоминания уже извлечены из текста. В первом этапе система определяет упоминания и формирует их векторные представления. Вместо этого формируются векторные представления предоставленных упоминаний и устанавливается значение оценок $s_m(i)$ того, что они являются упоминаниями равные 1. Архитектура модели изображена на рисунке 4.2.

На выход Bi-LSTM накладывается маска, определяющая положение извлечённых упоминаний. Таким образом, вместо набора всевозможных кандидатов в упоминания, механизм формирования векторного представления упоминаний работает только со скрытыми состояниями Bi-LSTM сети, относящимися к предоставленным упоминаниям. *Mention score net* в этом случае не используется и ее выход $s_m(i)$ устанавливается равным единице.

Базовая модель e2e-coref была проверена на наборе данных ConLL-2012 [98], чтобы убедиться в воспроизводимости результатов и на наборе данных RuCor [107], чтобы сравнить модель с существующими работами для

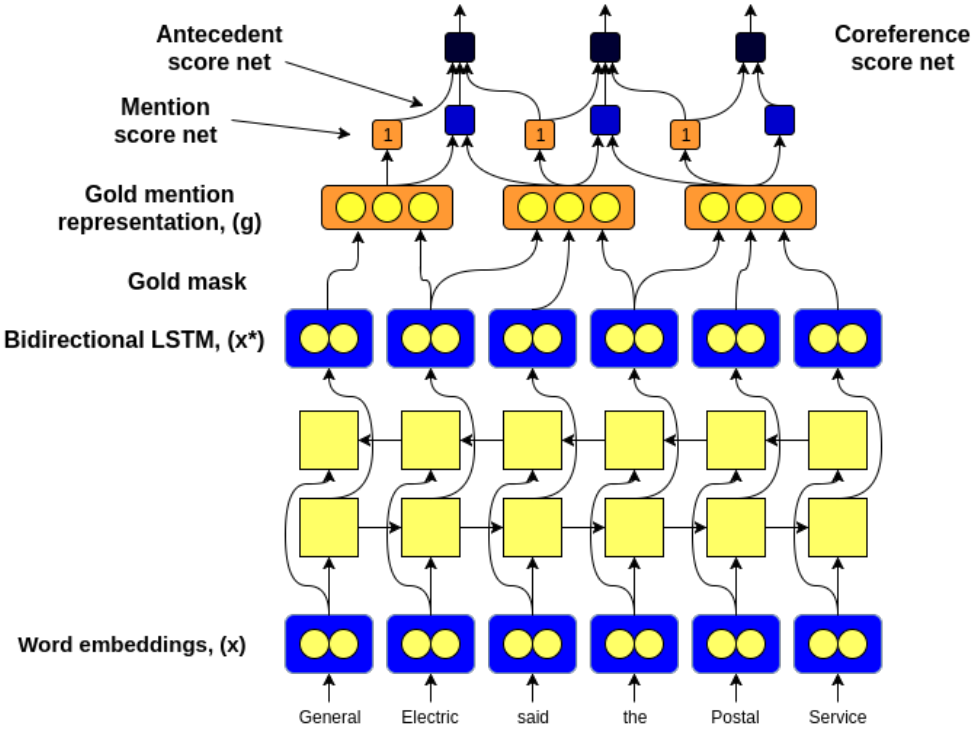


Рисунок 4.2 — Архитектура e2e-coref модели работающей с уже извлечёнными упоминаниями (gold mentions).

русского языка. В качестве векторных представлений слов для русского языка[5] использовались вектора FastText [4] обученные на русской части Википедии и корпусе Lenta.ru[10].

4.2.2 Базовая модель с ELMo

В базовой модели e2e-coref контекстно-зависимая информация о словах в упоминаниях строится только за счет рекуррентной Bi-LSTM сети. Для разрешения кореференции важно учитывать и весь контекст на уровне целого документа и локальные неоднозначности. Bi-LSTM сети приходится отвечать за оба типа контекстной зависимости. Для того, чтобы в некотором смысле снять нагрузку с Bi-LSTM можно добавить обработку контекстной информации с помощью контекстно-зависимых векторов от модели `ELMo` [6] в качестве входных наряду с векторами FastText [4]. Модель `ELMo` предобучена на большом

[5]Вектора FastText для русского языка выложены в библиотеке DeepPavlov `http://docs.deeppavlov.ai/en/master/features/pretrained_vectors.html#fasttext`. В этой работе использовались вектора с разбиением на слова с помощью функции wordpunct_tokenize из NLTK.

объеме текстов и состоит из двух слоев Bi-LSTM (4096 размер скрытого слоя) и сверточной сети, которая строит начальное представление слов из символов. ELMo [6] применяется на уровне предложений и позволяет лучше разрешать локальные неоднозначность внутри предложения. В экспериментах с моделью ELMo параметры самой языковой модели не обучались. Были обучены только веса для линейной комбинации выходов со всех слоев языковой модели ELMo. Такое обучение только весов линейной комбинации рекомендовано авторами ELMo [6]. Таким образом, в качестве входного вектора x_t для слова w_t использовалась вектор полученный объединением двух векторных представлений:

$$x_t = [FastText(w_t), ELMo(w_t)],$$

где

$$ELMo(w_t) = \gamma \sum_{j=0}^{2} \gamma_j H_{jt},$$

где H_{jt} - выходной вектор слоя j для позиции t, γ_j - вес слоя j в линейной комбинации, сумма γ_j равна 1 и γ коэффициент масштаба, слой 0 — сверточный, слой 1 — первая Bi-LSTM, слой 2 — вторая Bi-LSTM модели ELMo.

4.2.3 Базовая модель с RuBERT

В качестве контекстно-зависимых представлений слов можно использовать предобученные модели на основе трансформеров, вместо рекуррентных сетей. Для русского языка в качестве такой модели можно взять RuBERT. Модель RuBERT состоит из 12 трансформер слоев с выходной размерностью каждого равной 768. Аналогично тому как использовалась модель ELMo, можно брать линейную комбинацию выходов из слоев модели RuBERT. Входной вектор признаков для модели для разрешения кореференции становится следующим:

$$x_t = [FastText(w_t), RuBERT(w_t)].$$

В ходе начальных экспериментов были опробованы выходы с 1-6-12 и с 10-11-12 слоев для линейной комбинации. Было выяснено, что выходы с 1-6-12 слоев показывали стабильно лучшие значения метрик на валидационных наборах данных.

4.3 Результаты экспериментов

Первые эксперименты для русского языка были проведены на наборе данных RuCor [107] для того, чтобы сравниться с результатами полученными другими авторами ранее и проверить эффективность выбранных подходов. В таблицах 11 и 12 приведены результаты для базовой модели (Baseline). Базовая модель превзошла по всем значениям метрик ранее опубликованные результаты в работах Сысоева [114] и Толдовой [119]. На наборе данных RuCor также был получен результат для базовой модели с использованием контекстно-зависимых векторных представлений слов. В качестве источников контекстно-зависимых векторных представлений слов были использованы предобученные модели для русского языка `ELMo`[6] [6]. Эта модель позволила улучшить результаты базовой модели на 4 пункта F-1. Все последующие эксперименты по улучшению базовой модели с помощью новых языковых моделей проводились на новом наборе данных AnCor [109].

Таблица 11 — Результаты на наборе данных для разрешения кореференции RuCor [107], упоминания выделены из текста (gold mentions).

Модель	muc	$bcube$	$ceafe$	$avg.F_1$
Сысоев [114]	69.28	63.12	55.33	62.58
Толдова [119]	70.25	60.14	-	-
Baseline	**86.08**	**71.09**	**70.25**	**75.81**
Baseline + ELMo	**90.54**	**79.71**	**67.81**	**79.36**

Таблица 12 — Результаты на наборе данных для разрешения кореференции RuCor [107], текст без выделенных упоминаний (full).

Model	muc	$bcube$	$ceafe$	$avg.F_1$
Сысоев [114]	41.90	34.30	29.06	35.10
Baseline + ELMo	67.26	52.29	53.18	57.58

Для задачи разрешения кореференции в качестве базовой модели (Baseline в таблицах 13 и 14) была взята модель e2e-coref с контекстно-зависимыми векторными представлениями слов [117; 118]. Ранее на наборе данных RuCor [107]

[6]Модель `ELMo` для русского языка выложена в библиотеке DeepPavlov http://docs.deeppavlov.ai/en/master/features/pretrained_vectors.html#elmo.

Таблица 13 — Результаты на наборе данных для разрешения кореференции AnCor [109], упоминания выделены из текста (gold mentions).

Модель	muc	$bcube$	$ceafe$	$avg.F_1$
Baseline + ELMo	90.22	83.41	59.44	77.69
Baseline + ELMo + RuCor	**91.51**	84.16	61.33	79.01
Baseline + RuBERT(1-6-12)	91.04	84.38	63.07	79.50
Baseline + RuBERT(1-6-12) + RuCor	91.47	**84.49**	**63.81**	**79.92**

Таблица 14 — Результаты на наборе данных для разрешения кореференции AnCor [109], текст без выделенных упоминаний (full).

Модель	muc	$bcube$	$ceafe$	$avg.F_1$
Baseline + ELMo	50.29	48.89	46.99	51.72
Baseline + RuBERT(1-6-12)	60.95	51.08	49.24	53.76
Baseline + ELMo + RuCor	65.01	52.67	50.19	55.96
Baseline + RuBERT(1-6-12) + RuCor	**66.74**	**54.88**	**51.72**	**57.78**

была показана ее эффективность по сравнению с базовой моделью (таблицы 11 и 12). В качестве источников контекстно-зависимых векторных представлений слов помимо `ELMo` были проведены эксперименты с моделью `RuBERT`, описанной в главе 2.

Результаты по наборам данных AnCor и RuCor были получены путем усреднения метрик на 10 запусках эксперимента (10-folds). Модель *Baseline + RuBERT (1-6-12)* с признаками из 1-6-12 слоев `RuBERT` показала более высокие значения метрик, чем *RuBERT (10-11-12)* в предварительных экспериментах, поэтому далее использовался только *RuBERT (1-6-12)*. В ряде экспериментов с набором данных AnCor также использовался набор данных RuCor в качестве источника дополнительных обучающих данных (*+ RuCor* в таблицах 13 и 14).

Модели с использованием `RuBERT` показывают более высокие результаты на задаче разрешения кореференции, чем модели на основе `ELMo`. Проверялась только модель `RuBERT`, так как домен данных на которых была обучена модель `RuBERT` (Википедия и новости) более близок к домену данных в AnCor, чем домен разговорного `RuBERT`.

В таблицах 15, 16, 17, 18 приведены официальные результаты соревнования по разрешению кореференции и анафоры «Dialogue Evaluation 2019» на наборе данных AnCor [109]. Для задачи разрешения кореференции с вы-

деленными упоминаниями и для задачи разрешения анафоры модель *Baseline + RuBERT (1-6-12) + RuCor* с отрывом обошла решения других команд (76.24 $avg.F_1$ для кореференции и 83.7(69.4) $F_1(micro, strong)$ для анафоры). Для задачи разрешения кореференции без выделенных упоминаний предложенная модель показала близкие значения метрик с лучшей системой (SagTeam, таблица 16). К сожалению, только отчет команды Etap [120] был опубликован. Их решение основано на правилах поверх выдачи синтаксического и морфологического анализа системы ETAP-4[7]. Так как то, на чем основаны решения других команд неизвестно, глубокий анализ результатов невозможен.

Таблица 15 — Результаты соревнования по разрешению кореференции и анафоры Dialogue Evaluation 2019, AnCor. Разрешение кореференции, упоминания выделены из текста (gold mentions). Результаты на тестовом наборе данных.

Команда / Модель	muc	$bcube$	$ceafe$	$avg.F_1$
Baseline + RuBERT(1-6-12) + RuCor	**82.62**	**73.95**	**72.14**	**76.24**
Legacy	75.83	66.16	64.84	68.94
MorphoBabushka	61.36	53.39	51.95	55.57

Таблица 16 — Результаты соревнования по разрешению кореференции и анафоры Dialogue Evaluation 2019, AnCor. Разрешение кореференции, текст без выделенных упоминаний (full). Результаты на тестовом наборе данных.

Команда / Модель	muc	$bcube$	$ceafe$	$avg.F_1$
SagTeam	**62.23**	52.79	**52.29**	**55.77**
Baseline + RuBERT(1-6-12) + RuCor	62.06	**53.54**	51.46	55.68

В таблице 18 приведена официальная таблица с результатами соревнования по разрешению кореференции и анафоры «Dialogue Evaluation 2019», в которой организаторы объединили решения, работающие без выделенных упоминаний (full) и с ними (gold). По итогам соревнования, наше решение заняло первое место для задачи разрешения кореференции и для задачи разрешения анафоры (таблица 17). Работа по участию в соревновании велась в команде и результаты работы опубликованы в совместной статье «Sentence Level Representation and Language Models in the Task of Coreference Resolution for Russian» [20]. Состав команды: Ле Тхе Ань, Петров Максим, Куратов Юрий,

[7] http://proling.iitp.ru/etap4download

Таблица 17 — Результаты соревнования по разрешению кореференции и анафоры Dialogue Evaluation 2019, AnCor. Разрешение анафоры. Результаты на тестовом наборе данных. Полную таблицу результатов со всеми метриками можно найти в отчете организаторов соревнования [109]. Участники соревнования не сообщили в каком из режимов (gold mentions или full) были получены результаты.

Команда / Модель	$F_1(micro, soft)$	$F_1(micro, strong)$
Baseline + RuBERT(1-6-12) + RuCor [gold]	**91.2**	**83.7**
Baseline + RuBERT(1-6-12) + RuCor [full]	**77.8**	**69.4**
Legacy	73.2	61.0
MorphoBabushka	63.2	39.0
Meanotek	62.9	47.0
Etap [120]	62.9	46.9

Бурцев Михаил. Все члены команды принимали участие в обсуждении идей экспериментов, Ле Тхе Ань проводил эксперименты с поиском кореферентных предложений, Петров Максим и Куратов Юрий проводили эксперименты с моделями, основанными на языковых моделях `ELMo` и `RuBERT`, Петров Максим и Куратов Юрий собирали итоговые варианты решений для отправки организаторам соревнования по разрешению кореференции и анафоры «Dialogue Evaluation 2019».

Таблица 18 — Официальная таблица с результатами соревнования по разрешению кореференции и анафоры Dialogue Evaluation 2019, AnCor. Разрешение кореференции. Результаты на тестовом наборе данных.

Команда / Модель	muc	$bcube$	$ceafe$	$avg.F_1$
Baseline + RuBERT(1-6-12) + RuCor [gold]	**82.62**	**73.95**	**72.14**	**76.24**
Legacy	75.83	66.16	64.84	68.94
SagTeam	62.23	52.79	52.29	55.77
Baseline + RuBERT(1-6-12) + RuCor [full]	62.06	**53.54**	51.46	55.68
MorphoBabushka	61.36	53.39	51.95	55.57
Julia Serebrennikova	48.07	34.7	38.48	40.42

Как показали результаты для задачи разрешения кореференции, предобученные языковые модели `ELMo` и `RuBERT` позволяют значительно улучшить

качество систем для разрешения кореференции и анафоры для русского языка. Модели на основе `RuBERT` на данный момент являются наилучшими по метрикам *CoNLL-F-1* из опубликованных. Разработанные системы оказались заняли первые места на соревновании «Dialogue Evaluation 2019» по разрешению кореференции и анафоры. Также было показано, что дополнительные обучающие данные (RuCor [107]) позволяют улучшить результаты на наборе данных AnCor [109].

4.4 Новые модели, которые появились после экспериментов, проведенных в данной работе.

В работе «BERT for Coreference Resolution: Baselines and Analysis» [121] использовалась модель `BERT` и модель e2e-coref [117; 118] для разрешения кореференции на наборе данных CoNLL-2012 [98]. В отличие от экспериментов описанных в разделе 4.2.3 веса модели `BERT` обучались вместе со всей моделью разрешения кореференции и был убрана двухслойная Bi-LSTM. Представления для упоминаний строились прямо из выходов модели `BERT`. Также авторами были проведены эксперименты не только с `BERT-Base` версией модели со 110 миллионами параметров, но и с `BERT-Large` версией (340 миллионов параметров). `BERT-Base` модель позволила улучшить результат модели e2e-coref [117; 118] на 0,9 F-1, а `BERT-Large` на 3,9 F-1.

В работе «SpanBERT: Improving Pre-training by Representing and Predicting Spans» [122] была предложена модель `SpanBERT`, которая отличается от модели `BERT` задачами, на которых происходило предобучение. Модель `BERT` предобучалась на двух задачах: MLM — предсказание скрытого слова и NSP — определение того, следует ли второй фрагмент текста за первым. Модель `SpanBERT` была предобучена тоже на двух задачах: MLM был изменен так, чтобы скрывать не один токен, а несколько подряд идущих токенов (больше токенов, чем одно слово как в whole-word masking, в котором скрываются только токены на которые разбивается одно слово) и SBO (Span Boundary Objective) — предсказание скрытого фрагмента текста по токенам на границах этого фрагмента. Такое предобучение предсказывать скрытый фрагмент позволяет модели `SpanBERT` выучить эффективное представление для фрагментов

текстов, например, упоминаний для задачи разрешения кореференции или ответов для задачи поиска ответа на вопрос в контексте (SQuAD [73]). Модель `SpanBERT` позволила улучшила результат модели e2e-coref [117; 118] на 6,6 F-1 на наборе данных CoNLL-2012 [98].

Основные результаты и выводы главы 4:

- Адаптированные для русского языка базовые модели на основе работ «End-to-End Neural Coreference Resolution» [117] и «Higher-order Coreference Resolution with Coarse-to-fine Inference» [118] позволили улучшить ранее опубликованные результаты на наборе данных RuCor [107];
- Добавление размеченных данных из RuCor [107] позволяет улучшить результаты работы моделей на наборе данных AnCor [109];
- Модели, основанные на обученной в рамках работы над диссертации `RuBERT`, заняли первые места по разрешению кореференции и по разрешению анафоры для русского языка на соревновании «Dialogue Evaluation 2019» [109];
- Результаты полученные в этой главе опубликованы в работе «Sentence Level Representation and Language Models in the Task of Coreference Resolution for Russian» [20].

Глава 5. Вопросно-ответные системы и языковые модели

Вопросно-ответные системы используются для формирования ответа на вопрос пользователя в текстовой форме. Вопросно-ответные системы можно разделить на два основных типа: основанные на информационном поиске (information retrieval) и основанные на структурированных базах знаний. Если тематика вопросов ограничена, то под нужный домен могут быть построены базы знаний (вручную или полуавтоматически). Построить базы знаний с охватом всех возможных тематик уже гораздо сложнее. Из открытых баз знаний можно привести в пример DBPedia [123], WikiData [124], FreeBase [125] (стал частью WikiData) и коммерческие Amazon Evi[1], Google Knowledge Graph[2]. Поэтому для поиска ответа на вопросы неограниченные по тематике используются системы, основанные на информационном поиске.

Системы информационного поиска позволяют найти документы или фрагменты документов (сниппеты) релевантные вопросу, но не непосредственно сам ответ на вопрос. Пользователю приходится искать ответ самостоятельно среди подобранных документов. Появление моделей, извлекающих ответ из фрагмента текста [43; 126—128] и наборов данных SQuAD [73], MS MARCO [129], сделало возможным построение вопросно-ответных систем, работающих с коллекциями неструктурированных документов [130; 131].

Вопросно-ответные системы, основанные на информационном поиске можно подразделить на:

- поиск ответа на вопрос в коллекции пар вопрос-ответ, например, списки часто задаваемых вопросов (FAQ) на сайтах;
- поиск релевантного документа из коллекции, например, поисковые системы;
- поиск и извлечение ответа на вопрос в тексте, такая система может сформировать короткий ответ.

Две последние в этом списке системы могут быть объединены в одну (рисунок 5.1), которая сначала находит релевантные документы, а затем находит ответ среди них.

[1] https://en.wikipedia.org/wiki/Evi_(software)

[2] https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html

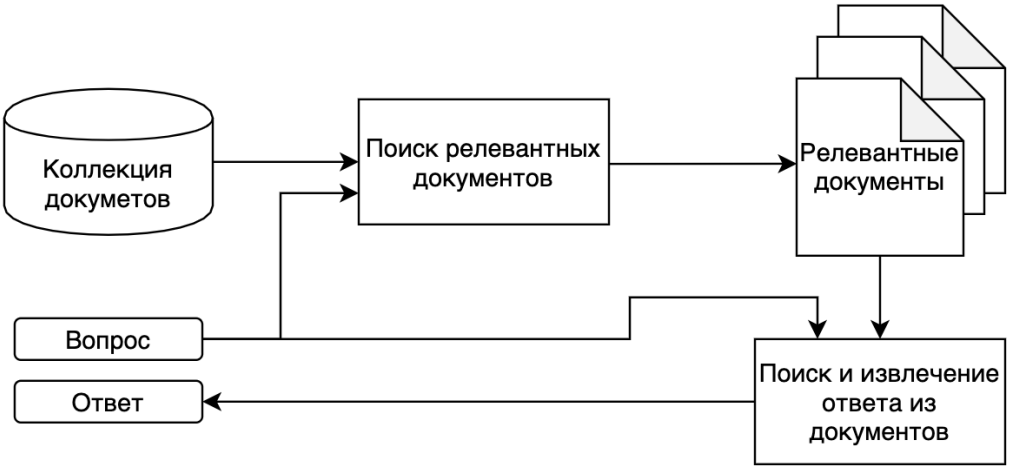


Рисунок 5.1 — Поиск и извлечение ответа из коллекции документов.

Системы, основанные на информационном поиске работают с текстовыми данными, в которых нет формальной структуры. Когда в знания структурированы (например, базы знаний), то задача поиска ответа на вопрос сводится к формированию структурированного запроса:

Вопрос: Кто автор «Ресторана в конце Вселенной»?
SPARQL запрос: SELECT ?answer WHERE { wd:Q578895 wdt:P50 ?answer }
ID ответа: wd:Q42
Ответ: Дуглас Адамс

Частью вопросно-ответной системы (рисунок 5.1) является модуль, отвечающий за поиск и извлечение ответа на вопрос в тексте. Далее в разделе 5.1 будет сформулирована постановка этой задачи и описаны модели, использованные для ее решения: R-Net и модель на основе `BERT`.

5.1 Поиск ответа на вопрос в тексте

Под поиском ответа на вопрос в тексте будем подразумевать следующую задачу: дан небольшой текст и вопрос к нему, система должна выбрать в качестве ответа непрерывный фрагмент из данного текста либо сообщить, что ответа в тексте нет.

Соединение пластмасс между собой может осуществляться механически (с помощью фигурных профилей, болтов, заклепок и т.д.), химически (склеиванием, растворением с последующим высыханием), термически (сваркой). Из перечисленных способов соединения **только при помощи сварки** можно получить соединение без инородных материалов, а также соединение, которое по свойствам и составу будет максимально приближено к основному материалу. Поэтому сварка пластмасс нашла применение при изготовлении конструкций, к которым предъявляются повышенные требования к герметичности, прочности и другим свойствам.
Вопрос: Каким образом можно получить соединение без инородных материалов?
Ответ: только при помощи сварки

Пример взят из набора данных SDSJ Задача B (подробнее про данные будет написано в разделе 5.1.3). В этом примере ответ на вопрос найти не так трудно, потому что многие слова в вопросе содержатся в исходном тексте в одном предложении. Сильное пересечение слов в вопросе и тексте является предметом критики наборов данных SQuAD и SDSJ Задача B и обусловлено методикой их построения [73; 86].

Не все примеры такие простые, многие из них требуют от системы умения делать логические выводы, использовать информацию из разных предложений, разрешать кореференцию, обладать общим знанием об устройстве мира (common sense) [73].

5.1.1 Описание подхода к поиску ответа на вопрос в тексте с использованием языковых моделей на базе архитектуры Трансформер

Как и для других задач обработки естественного языка, языковые модели на базе архитектуры Трансформер могут быть использованы для поиска ответа на вопрос в тексте. Языковая модель может быть только источником признаков, или также быть дообучена на решение нужной задачи.

Так как в нашей постановке задачи (раздел 5.1) ответ является непрерывным отрывком из текста, то он может быть однозначно задан позициями его начала и конца. Таким образом, задачу поиска ответа на вопрос в тексте можно свести к двум задачам классификации: является ли слово началом ответа и является ли слово концом ответа.

$$\mathcal{L}(\Theta, w_{st}, w_{end}) = -\sum_k (\log p_{st}(y_k^{st}|C,Q;\Theta,w_{st}) + \log p_{end}(y_k^{end}|C,Q;\Theta,w_{end})), \quad (5.1)$$

где C - текст, Q - вопрос, Θ - параметры языковой модели, w_{st} и w_{end} - обучаемые параметры для предсказания позиции начала и конца ответа соответственно, y_k^{st} и y_k^{end} - позиции начала и конца ответа для примера с индексом k. Вероятности p_{st} и p_{end} при использовании модели `BERT` определяются следующим образом:

$$p^{st}(y) = softmax([\langle w_{st}, BERT_0^N \rangle, \langle w_{st}, BERT_1^N \rangle, \ldots, \langle w_{st}, BERT_L^N \rangle]),$$

$$p^{end}(y) = softmax([\langle w_{end}, BERT_0^N \rangle, \langle w_{end}, BERT_1^N \rangle, \ldots, \langle w_{end}, BERT_L^N \rangle]),$$

где L — число токенов во входной последовательности, $\langle \cdot, \cdot \rangle$ — скалярное произведение, $BERT_i^N$ — выход с последнего слоя N для i-ого токена во входной последовательности.

Вопрос Q подается в качестве первого сегмента, а текст C подается как второй сегмент. Сегменты разделены специальным токеном $[SEP]$, также в самое начало последовательности вставляется токен $[CLS]$ (рисунок 5.2). Если в тексте C нет ответа на вопрос Q, то модель должна выбрать токен $[CLS]$ в качестве правильного ответа. В наборе данных SQuAD [73] такого не встречается, но может возникнуть из-за обработки входного текста — обрезания его по длине.

5.1.2 Базовая модель на основе R-Net

Модель R-Net [43] была разработана исследователями из Microsoft и появилась раньше, чем вышла статья «Attention is All You Need» [33] в которой была описана архитектура Трансформер.

Основной идей всех моделей для поиска ответа на вопрос в тексте для набора данных SQuAD [73] было построение совместных представлений для вопроса и слов в тексте и потом выбор позиций начала и конца ответа.

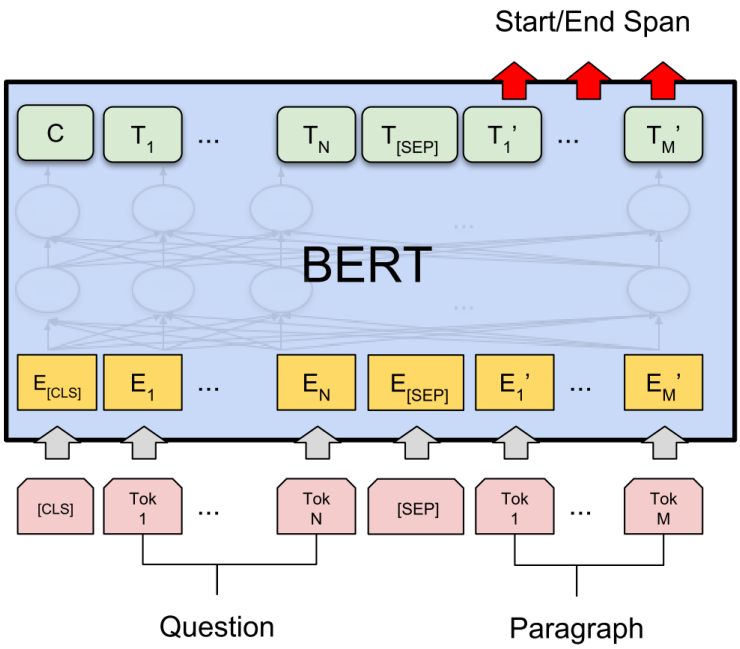


Рисунок 5.2 — Применение языковой модели на базе архитектуры Трансформер для задачи поиска ответа на вопрос в тексте. Рисунок взят из оригинальной статьи BERT [8].

Архитектуру модели R-Net можно разбить на 5 частей изображенных на рисунке 5.3:

1. Построение векторных представлений слов текста и вопроса. Word & Character.
2. Построение контекстно-зависимых векторных представлений слов текста и вопроса (независимое друг от друга). Question & Passage.
3. Добавление информации о вопросе в представления слов текста. Question-Passage Matching.
4. Аггрегация информации о всем тексте для каждого слова. Passage Self-Matching.
5. Предсказание позиций начала и конца ответа. Answer Prediction.

Векторные представления слов строятся из предобученных векторных представлений (GloVe [3]) и последнего скрытого состояния двунаправленной рекуррентной сети с GRU-ячейками [29], которой на вход подаются векторные представления символов слов:

$$e_t^Q = [E_{Q_t}^{word}, BiGRU_{char}(E_{Q_{t,0}}^{char}, \ldots, E_{Q_{t,L_t}}^{char})],$$

$$e_t^P = [E_{P_t}^{word}, BiGRU_{char}(E_{P_{t,0}}^{char}, \ldots, E_{P_{t,L_t}}^{char})],$$

где e_t^Q — итоговое векторное представление слова t в вопросе,e_t^Q — в тексте, $Q_{t,i}$ — i-ый символ в слове t длины L_t в вопросе, E^{word} и E^{char} — предобученные матрицы векторных представлений слов.

Затем, полученные векторные представления слов подаются в трехслойную Bi-GRU выходы для каждого слоя из которой формируют контекстно-зависимые представлений слов в вопросе и в контексте (Question & Passage Encoding на рисунке 5.3):

$$u^P = BiGRU_{enc}(e_0^P, \ldots, e_{L_P}^P),$$

$$u^Q = BiGRU_{enc}(e_0^Q, \ldots, e_{L_Q}^Q).$$

Далее, с помощью механизма внимания добавляется информация о вопросе к каждому из представлений u^P слов в тексте. Механизм внимания позволяет для каждого слова t в тексте сформировать вектор c_t, в котором агрегируется информация из вопроса, релевантная именно этому слову (Question-Passage Matching на рисунке 5.3).

Также есть управляющий механизм (gate), который определяет какую информация в получившемся векторе $[u_t^P, c_t]$ нужно передавать на следующие слои:

$$[u_t^P, c_t]^* = g_t \odot [u_t^P, c_t]$$

$$g_t = \sigma(W_{gate}[u_t^P, c_t]).$$

Этот управляющий механизм помогает, например, определять какие слова в тексте релевантны вопросу. Затем, полученные вектора подаются на вход однослойной Bi-GRU:

$$v_t^P = BiGRU_{QP}([u_t^P, c_t]^*).$$

Этот шаг немного отличается от оригинальной архитектуры R-Net и реализован в библиотеке DeepPavlov[3].

Затем, механизм внимания примется для каждого слова в тексте, чтобы получить агрегированную информацию для этого слова по всему тексту. Также используется управляющий механизм и Bi-GRU. Этот этап (Passage Self-Matching) устроен также как и предыдущий, только механизм внимания работает на вектора самого текста v^P.

[3] http://docs.deeppavlov.ai/en/master/features/models/squad.html#r-net

Чтобы определить позиции начала и конца ответа используется Pointer Network [132], которая с помощью механизма внимания на первом рекуррентном шаге выбирает позицию начала ответа, а на втором — позицию конца. Начальное скрытое состояние Pointer Nerwork инициализируется средневзвешенной суммой векторных представлений слов в вопросе u^Q, веса определяются механизмом внимания.

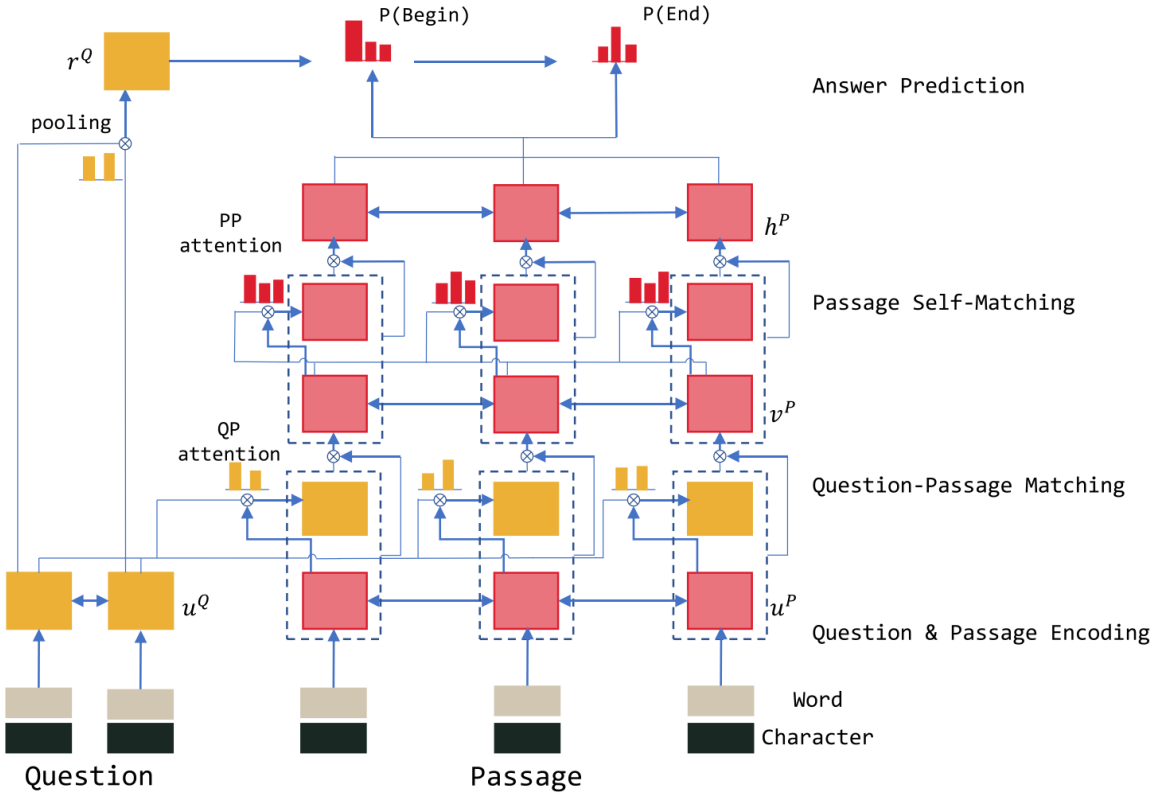


Рисунок 5.3 — Архитектура модели R-Net [43]. Рисунок взят из технического отчета https://www.microsoft.com/en-us/research/publication/mcr/.

Адаптация модели R-Net для русского языка. Для того, чтобы использовать модель R-Net для русского языка достаточно изменить используемые векторные представления слов и символов. В качестве векторных представления слов были взяты FastText [4] вектора, обученные на Википедии и новостных данных Lenta.ru[4]. Эти векторные представления выложены в библиотеке DeepPavlov[5].

Векторные представления для символов E^{char} были построены из векторных представлений слов E^{word} — брались вектора для всех слов в которых есть

[4] https://github.com/yutkin/Lenta.Ru-News-Dataset

[5] http://docs.deeppavlov.ai/en/master/features/pretrained_vectors.html#fasttext

данный символ и усреднялись:

$$E_c^{char} = \frac{\sum_{\{w \in W | c \in w\}} E_w^{word}}{|\{w \in W | c \in w\}|}.$$

5.1.3 Описание данных

SDSJ Задание B: SberSQuAD В рамках соревнования Sberbank Data Science Journey[6] 2017 была представлено две задачи: **A** (на классификацию) и **B** (на поиск ответа на вопрос в тексте). Также набор данных SDSJ Задача **B** известен как SberQuAD [86].

Задание B идейно основано на наборе данных собранных Университетом Стэнфорда (Stanford Question Answering Dataset, SQuAD [73]) для английского языка. Организаторы соревнования SDSJ собрали около 50,000 вопросов к текстам с ответами (вместе тренировочные и валидационные данные) на русском языке. Ответ на вопрос всегда является непрерывным отрывком из контекста.

Создание набора данных SQuAD [73] дало старт появлению сложных и эффективных нейросетевых архитектур, таких как BiDAF [127], R-NET [43], Mnemonic Reader [128], специализирующихся на этой задаче. Эти модели основаны на механизмах внимания [37; 38] и построении совместного векторного представления вопроса и текста (параграфа, контекста). С появлением архитектуры Трансформер [33], состоящей только из полносвязных слоев и механизма внимания, все лучшие модели стали использовать Трансформер вместо рекуррентных и сверточных сетей.

Во время подготовки данных для экспериментов были обнаружены ошибки в наборе данных, например, ответ мог не находиться как подстрока в тексте из-за опечаток в ответах или разной токенизации. Также встречались ответы формата *да/нет*, что не соответствует постановке задачи. Все несоответствия были откорректированы (или удалены из набора данных, если нет очевидного исправления), и полученный набор данных был выложен в открытый доступ[7] для возможности воспроизведения результатов полученных в разделе 5.1.5. В [86] так же говорится о том, что в исходном наборе данных содержатся ошибки.

[6] https://sdsj.sberbank.ai/

[7] http://files.deeppavlov.ai/datasets/sber_squad_clean-v1.1.tar.gz

5.1.4 Метрики

EM — точное совпадение (exact match), доля ответов системы, которые полностью совпадают с одним из правильных ответов (с точностью до пунктуации и регистра).

F-1 — средняя доля общих слов в ответе системы $A_{predicted}$ и в правильном ответе A_{true}. Сначала ответы нормализуются — убирается пунктуация, приводятся к нижнему регистру и разбиваются на слова. Затем вычисляется точность и полнота для одного примера:

$$P(A_{predicted}, A_{true}) = \frac{|A_{predicted} \cap A_{true}|}{|A_{predicted}|},$$

$$R(A_{predicted}, A_{true}) = \frac{|A_{predicted} \cap A_{true}|}{|A_{true}|}$$

и F-1 мера по стандартной формуле:

$$F\text{-}1 = \frac{2 \cdot P \cdot R}{P + R}.$$

Так как может быть несколько правильных ответов на один вопрос, то берется максимальное значение меры F-1 среди всех правильных ответов на этот вопрос.

5.1.5 Результаты

Для русского языка были обучены модели на основе многоязычного `BERT` и на основе `RuBERT` на наборе данных SDSJ Задани B (SberQuAD). Также сравнены с базовой моделью на основе R-Net [43]. Результаты экспериментов приведены в таблице 19.

Модель на основе `BERT` позволила повысить качество базовой модели на основе R-Net [43] на 3,35 F-1 (83,39 F-1) и 3,73 EM (64,35 EM). Модель `RuBERT` адаптированная специально для русского языка увеличила значения метрик еще на 1,21 F-1 (84,6 F-1) и 1,95 EM (66,30 EM).

Таблица 19 — Результаты поиска ответа на вопрос в тексте. SDSJ Задание B. Качество моделей оценивалась на валидационном наборе данных (публичный лидерборд). Значения получены усреднением результатов 5 запусков экспериментов.

Модель	F-1 (dev)	EM (dev)
R-Net [43]	80.04	60.62
Многоязычный BERT	83.39 ± 0.08	64.35 ± 0.39
RuBERT	$\mathbf{84.60} \pm 0.11$	$\mathbf{66.30} \pm 0.24$

Основные результаты и выводы главы 5:

- Реализована базовая модель поиска ответа на вопрос в тексте на основе R-Net [43] и адаптирована для русского языка.
- Реализована модель поиска ответа на вопрос в тексте на основе `RuBERT`.
- Оценка качества разработанных решений показала, что модель `RuBERT` специально обученная для русского языка показывает лучшие результаты, чем многоязычный `BERT` на задаче поиска ответа на вопрос в тексте.
- Получены лучшие опубликованные результаты на наборе данных для русского языка SDSJ Задача B: 84,60 F-1 и 66,30 EM.
- Все модели выложены в публичный доступ в библиотеке DeepPavlov[8].
- Результаты опубликованы в работе «Adaptation of deep bidirectional multilingual transformers for russian language» [19].
- Были найдены и исправлены ошибки в наборе данных SDSJ Задача B, исправленная версия выложена в открытый доступ.

[8] http://docs.deeppavlov.ai/en/master/features/models/squad.html

Заключение

Основные результаты данной диссертации заключаются в следующем.

1. Предложен метод переноса знаний с обученных языковых моделей `BERT` и показано, что предложенный метод переноса знаний позволяет ускорить процесс предобучения языковых моделей.
2. Частью предложенного метода переноса знаний является пересборка словаря под новый язык или домен. Пересобранные словари позволяют уменьшить длины входных последовательностей (примерно в 1,6 раз для русского языка), а значит ускорить работу моделей и уменьшить требования к доступной оперативной или видеопамяти.
3. Предложенный метод переноса знаний применен для предобучения языко-специфичных `RuBERT` для русского языка и `Славянский BERT` для болгарского, чешского, польского и русского языков. Также, предложенный метод переноса знаний применен для предобучения моделей разговорного домена для английского и русского языков.
4. На задачах классификации текста, разметки последовательности и поиска ответа на вопрос в тексте показано, что языко-специфичные модели демонстрируют лучшие результаты, чем многоязычные модели для заданного языка (улучшение от 1,2 F-1 до 6,5 F-1 на разных задачах).
5. На задачах классификации текста показано, что предобученные языковые модели на данных схожего домена демонстрируют лучшие значения метрик, чем модели общего домена (улучшение от 0,8 F-1 до 3,67 F-1 на разных задачах).
6. Обученные в рамках работы над диссертацией языко- и доменно- специфичные модели позволили улучшить ранее опубликованные результаты для разных задач обработки естественного языка (классификации, распознавания именованных сущностей, разрешения кореференции и анафоры, поиска ответа на вопрос в тексте) и для разных наборов данных (RuSentiment, ParaPhraser, Collection-3, BSNLP-2019, RuCor, AnCor, SDSJ Task B (SberQuAD)).

7. Все предобученные языковые модели выложены в публичный доступ в библиотеке DeepPavlov, как и большинство моделей обученных на целевых задачах.

В качестве рекомендаций по **применению полученных результатов** хотелось бы отметить, что предложенный метод переноса знаний позволяет сократить время необходимое для предобучения модели (инициализированной случайными параметрами) примерно на 5 дней (на DGX-1 с 8 P-100 16 Гб). Таким образом, перенос знаний помогает дообучить модели в условии ограниченных временных ресурсов. Предложенный метод переноса знаний может быть применен к любым доменам и к любым языкам из тех, на которых была предобучена модель `многоязычный BERT`. Предложенный метод переноса знаний может быть применен для предобучения других языковых моделей (например GPT [7], ALBERT [64], XLNet [61]).

Рекомендации касательно обученных в данной работе языковых моделей: следует использовать модель `RuBERT` для решения задач на русском языке вместо модели `многоязычный BERT`. Также рекомендуется использовать модель `Славянский BERT` для решения задач на болгарском, чешском, польском языках вместо модели `многоязычный BERT`. В случае, если данные решаемой задачи имеют разговорную структуру или содержат неформальный язык, то стоит использовать модели `разговорный RuBERT` для русского и `разговорный BERT` для английского языка. Домен используемой предобученной языковой модели важен, поэтому при наличии достаточно больших объемов текстовых данных нужного домена стоит дообучать языковую модель на этих данных методом, предложенным в этой работе. В открытый доступ выложен целый ряд моделей, решающих различные задачи обработки естественного языка. Эти модели могут быть использованы как есть, либо быть обучены на нужных данных с помощью библиотеки DeepPavlov.

Далее обозначены возможные **перспективы разработки темы диссертации**.

Предложенный в данной работе метод переноса знаний использует пересборку словарей и инициализацию векторных представлений для новых саботокенов, при этом все остальные параметры языковой модели не изменяются. Разработка новых методов для более эффективной инициализации остальных параметров модели, может позволить ускорить процесс дообучения языковой модели.

Большинство языковых моделей на базе архитектуры Трансформер работает с представлением слов в виде сабтокенов. Сабтокены получаются с помощью алгоритма BPE, который может быть не лучшим решением. Например, одно и тоже слово с разной капитализацией или числа близкие по значению могут быть разбиты алгоритмом на сильно отличающиеся части. Алгоритм BPE универсален и может работать с любыми типами последовательностей — это означает, что особенности естественного языка в нем не учтены. Есть работы, которые решают некоторые из проблем связанных с BPE токенизацией [133; 134], но больше исследований может быть проведено в этом направлении.

Также перспективным направлением является применение предложенного метода переноса знаний для обучения новых языковых моделей для русского языка на основе других обученных моделей. Могут быть обучены языковые модели бóльших размеров для улучшения качества на целевых задачах и также модели малых размеров (дистилляция знаний [135], квантизация, разреженные сети), для снижения требований к вычислительным ресурсам и более эффективного применения в практических приложениях.

Список сокращений и условных обозначений

BERT Bidirectional Encoder Representations from Transformers, представления из двунаправленных кодировщиков с архитектурой Трансформер

Bg Болгария, болгарский

BPE Byte Pair Encoding, кодирование наиболее встречаемой пары байт

CBOW Continuous Bag-of-Words, непрерывный мешок слов

CRF Conditional Random Field, условное случайное поле

Cz Чехия, чешский

ELMo Embeddings from Language Models, векторные представления из языковых моделей

EM Exact Match, точное совпадение

FAQ Frequently Asked Questions, часто задаваемые вопросы

FC Fully Connected, полносвязный слой

FFNN Feedforward Neural Network, нейронная сеть прямого распространения

GPT Generative Pre-Training, генеративное предобучение

GRU Gated Recurrent Unit, управляемый рекуррентный блок

OOV Out-of-vocabulary, слово не из словаря

Pl Польша, польский

RNN Recurrent Neural Network, рекуррентная нейронная сеть

Ru Россия, русский

LSTM Long Short-Term Memory, долгая краткосрочная память

MLM Masked Language Modeling, маскированное языковое моделирование

NSP Next Sentence Prediction, предсказание следующего предложения

Словарь терминов

Векторное представление слова : word embedding, представление слова в виде вещественного вектора фиксированной длины

N-грамма : последовательность из N элементов, в контексте данной работы элементами могут быть слова, токены, символы

Униграмма : последовательность из одного элемента

Биграмма : последовательность из двух элемента

Токен : единица текста, может быть словом, символьной N-граммой

Токенизация : процесс разбиения текста на токены. Например, разбиение текста по пробелам и символам пунктуации

Сабтокен : subtoken, subword unit, подслово, часть слова. В данной работе понятие сабтокен используется для обозначения результата токенизации текста с помощью алгоритма BPE

Языковая модель : вероятностная модель распределения следующего слова в тексте по N предыдущим $P(w_{N+1}|w_1, w_2, ..., w_N)$. Также, моделирует вероятность встретить данную последовательность из N подряд идущих слов $P(w_1, w_2, \ldots, w_N)$. В данной работе под языковой моделью понимается более общий класс моделей $P(w_i|w_{i-c_L}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+c_R})$, где c_L и c_R — длины левого и правого контекста для слова на позиции i

Маскированная языковая модель : вероятностная модель распределения слова в позиции i по полному левому и правому контексту $P(w_i|w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_N)$

Словарь : набор слов (токенов, сабтокенов). Языковой моделью строится распределение вероятностей элементов словаря.

Корпус : коллекция документов, текстов

Дообучение : fine-tuning, обучение модели на новом наборе данных (возможно на другой задаче), которая была ранее уже обучена на другом наборе данных

Предобучение : процесс предварительного обучения модели, который применяется перед обучением модели на целевом наборе данных (или задаче)

Перенос знаний : transfer learning, использование знаний, полученных во время обучения на одной задаче (и/или домене) для обучения модели на другой задаче (и/или домене). Более общее понятие, чем дообучение и предобучение.

Конкатенация : от англ. concatenate, операция объединения двух векторов (последовательностей) в один, в которой элементы вектора $b = [b_1, \ldots, b_{L_b}]$ добавляются в конец вектора $a = [a_1, \ldots, a_{L_a}]$:

$$[a, b] = [a_1, \ldots, a_{L_a}, b_1, \ldots, b_{L_b}],$$

где L_a и L_b — длины векторов a и b

Список литературы

1. Efficient Estimation of Word Representations in Vector Space / T. Mikolov, K. Chen, G. Corrado, J. Dean // 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings / ed. by Y. Bengio, Y. LeCun. — 2013. — URL: http://arxiv.org/abs/1301.3781.
2. Distributed Representations of Words and Phrases and their Compositionality / T. Mikolov [et al.] // Advances in Neural Information Processing Systems 26 / ed. by C. J. C. Burges [et al.]. — Curran Associates, Inc., 2013. — P. 3111—3119. — URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.
3. *Pennington, J.* GloVe: Global Vectors for Word Representation / J. Pennington, R. Socher, C. D. Manning // Empirical Methods in Natural Language Processing (EMNLP). — 2014. — P. 1532—1543. — URL: http://www.aclweb.org/anthology/D14-1162.
4. Enriching Word Vectors with Subword Information / P. Bojanowski, E. Grave, A. Joulin, T. Mikolov // Transactions of the Association for Computational Linguistics. — 2017. — Vol. 5. — P. 135—146. — URL: https://www.aclweb.org/anthology/Q17-1010.
5. *Dai, A. M.* Semi-supervised Sequence Learning / A. M. Dai, Q. V. Le // Advances in Neural Information Processing Systems 28 / ed. by C. Cortes [et al.]. — Curran Associates, Inc., 2015. — P. 3079—3087. — URL: http://papers.nips.cc/paper/5949-semi-supervised-sequence-learning.pdf.
6. Deep Contextualized Word Representations / M. Peters [et al.] // Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). — New Orleans, Louisiana : Association for Computational Linguistics, 06/2018. — P. 2227—2237. — URL: https://www.aclweb.org/anthology/N18-1202.
7. Improving language understanding with unsupervised learning : tech. rep. / A. Radford, K. Narasimhan, T. Salimans, I. Sutskever. — 2018.

8. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin, M.-W. Chang, K. Lee, K. Toutanova // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). — 2019. — P. 4171—4186.
9. Language models are unsupervised multitask learners / A. Radford [et al.]. —.
10. Language models are few-shot learners / T. B. Brown [et al.] // arXiv preprint arXiv:2005.14165. — 2020.
11. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism / M. Shoeybi [et al.] // arXiv preprint arXiv:1909.08053. — 2019.
12. Camembert: a tasty french language model / L. Martin [et al.] // arXiv preprint arXiv:1911.03894. — 2019.
13. FlauBERT: Unsupervised Language Model Pre-training for French / H. Le [et al.] // Proceedings of The 12th Language Resources and Evaluation Conference. — 2020. — P. 2479—2490.
14. *Beltagy, I.* SciBERT: A Pretrained Language Model for Scientific Text / I. Beltagy, K. Lo, A. Cohan // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). — Hong Kong, China : Association for Computational Linguistics, 11/2019. — P. 3615—3620. — URL: https://www.aclweb.org/anthology/D19-1371.
15. *Tran, K.* From english to foreign languages: Transferring pre-trained language models / K. Tran // arXiv preprint arXiv:2002.07306. — 2020.
16. *Lample, G.* Cross-lingual Language Model Pretraining / G. Lample, A. Conneau // Advances in Neural Information Processing Systems (NeurIPS). — 2019.
17. Unsupervised Cross-lingual Representation Learning at Scale / A. Conneau [et al.] // arXiv preprint arXiv:1911.02116. — 2019.
18. *Куратов, Ю. М.* Применение нейросетевых методов к задаче разрешения кореференции / Ю. М. Куратов // Материалы Международного молодежного научного форума «ЛОМОНОСОВ-2018». — 2018.

19. *Kuratov, Y.* Adaptation of deep bidirectional multilingual transformers for russian language / Y. Kuratov, M. Arkhipov // Computational Linguistics and Intellectual Technologies. International Conference "Dialogue 2019" Proceedings. — 2019. — P. 333—339.

20. Sentence Level Representation and Language Models in the Task of Coreference Resolution for Russian / T. A. Le, M. A. Petrov, Y. M. Kuratov, M. S. Burtsev // Computational Linguistics and Intellectual Technologies. International Conference "Dialogue 2019" Proceedings. — 2019. — P. 364—373.

21. DeepPavlov: Open-Source Library for Dialogue Systems / M. Burtsev [et al.] // Proceedings of ACL 2018, System Demonstrations. — 2018. — P. 122—127.

22. Tuning Multilingual Transformers for Language-Specific Named Entity Recognition / M. Arkhipov, M. Trofimova, Y. Kuratov, A. Sorokin // Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing. — Florence, Italy : Association for Computational Linguistics, 08/2019. — P. 89—93. — URL: https://www.aclweb.org/anthology/W19-3712.

23. *Murphy, K. P.* Machine learning: a probabilistic perspective / K. P. Murphy. — MIT press, 2012.

24. *Schütze, H.* Introduction to information retrieval. Vol. 39 / H. Schütze, C. D. Manning, P. Raghavan. — Cambridge University Press Cambridge, 2008.

25. *Xu, W.* Can artificial neural networks learn language models? / W. Xu, A. Rudnicky // Sixth international conference on spoken language processing. — 2000.

26. A neural probabilistic language model / Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin // Journal of machine learning research. — 2003. — Vol. 3, Feb. — P. 1137—1155.

27. *Mikolov, T.* Jan ˇCernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model / T. Mikolov, M. Karafiát, L. Burget // Eleventh annual conference of the international speech communication association. — 2010. — P. 1045—1048.

28. *Hochreiter, S.* Long short-term memory / S. Hochreiter, J. Schmidhuber // Neural computation. — 1997. — Vol. 9, no. 8. — P. 1735—1780.

29. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation / K. Cho [et al.] // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). — Doha, Qatar : Association for Computational Linguistics, 10/2014. — P. 1724—1734. — URL: https://www.aclweb.org/anthology/D14-1179.

30. *Zaremba, W.* Recurrent neural network regularization / W. Zaremba, I. Sutskever, O. Vinyals // arXiv preprint arXiv:1409.2329. — 2014.

31. Quasi-recurrent neural networks / J. Bradbury, S. Merity, C. Xiong, R. Socher // arXiv preprint arXiv:1611.01576. — 2016.

32. *Merity, S.* Regularizing and Optimizing LSTM Language Models / S. Merity, N. S. Keskar, R. Socher // International Conference on Learning Representations. — 2018. — URL: https://openreview.net/forum?id=SyyGPP0TZ.

33. Attention is All you Need / A. Vaswani [et al.] // Advances in Neural Information Processing Systems 30 / ed. by I. Guyon [et al.]. — Curran Associates, Inc., 2017. — P. 5998—6008. — URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

34. *Graves, A.* Generating sequences with recurrent neural networks / A. Graves // arXiv preprint arXiv:1308.0850. — 2013.

35. *Sutskever, I.* Sequence to Sequence Learning with Neural Networks / I. Sutskever, O. Vinyals, Q. V. Le // Advances in Neural Information Processing Systems 27 / ed. by Z. Ghahramani [et al.]. — Curran Associates, Inc., 2014. — P. 3104—3112. — URL: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

36. Convolutional Sequence to Sequence Learning / J. Gehring [et al.] // ICML. — 2017. — P. 1243—1252. — URL: http://proceedings.mlr.press/v70/gehring17a.html.

37. *Bahdanau, D.* Neural machine translation by jointly learning to align and translate / D. Bahdanau, K. Cho, Y. Bengio // arXiv preprint arXiv:1409.0473. — 2014.

38. *Luong, T.* Effective Approaches to Attention-based Neural Machine Translation / T. Luong, H. Pham, C. D. Manning // Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. — Lisbon, Portugal : Association for Computational Linguistics, 2015. — P. 1412—1421. — URL: https://www.aclweb.org/anthology/D15-1166.

39. Human behavior and the principle of least effort / G. K. Zipf [et al.]. — 1949.

40. *Powers, D. M. W.* Applications and Explanations of Zipf's Law / D. M. W. Powers // New Methods in Language Processing and Computational Natural Language Learning. — 1998. — URL: https://www.aclweb.org/anthology/W98-1218.

41. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation / W. Ling [et al.] // Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. — Lisbon, Portugal : Association for Computational Linguistics, 09/2015. — P. 1520—1530. — URL: https://www.aclweb.org/anthology/D15-1176.

42. *Luong, T.* Better Word Representations with Recursive Neural Networks for Morphology / T. Luong, R. Socher, C. Manning // Proceedings of the Seventeenth Conference on Computational Natural Language Learning. — Sofia, Bulgaria : Association for Computational Linguistics, 08/2013. — P. 104—113. — URL: https://www.aclweb.org/anthology/W13-3512.

43. Gated Self-Matching Networks for Reading Comprehension and Question Answering / W. Wang [et al.] // Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). — Vancouver, Canada : Association for Computational Linguistics, 07/2017. — P. 189—198. — URL: https://www.aclweb.org/anthology/P17-1018.

44. *Ma, X.* End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF / X. Ma, E. Hovy // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). — Berlin, Germany : Association for Computational Linguistics, 08/2016. — P. 1064—1074. — URL: https://www.aclweb.org/anthology/P16-1101.

45. *Sennrich, R.* Neural Machine Translation of Rare Words with Subword Units / R. Sennrich, B. Haddow, A. Birch // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). — Berlin, Germany : Association for Computational Linguistics, 08/2016. — P. 1715—1725. — URL: https://www.aclweb.org/anthology/P16-1162.

46. *Gage, P.* A new algorithm for data compression / P. Gage // C Users Journal. — 1994. — Vol. 12, no. 2. — P. 23—38.

47. Google's neural machine translation system: Bridging the gap between human and machine translation / Y. Wu [et al.] // arXiv preprint arXiv:1609.08144. — 2016.

48. Rich feature hierarchies for accurate object detection and semantic segmentation / R. Girshick, J. Donahue, T. Darrell, J. Malik // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2014. — P. 580—587.

49. Decaf: A deep convolutional activation feature for generic visual recognition / J. Donahue [et al.] // International conference on machine learning. — 2014. — P. 647—655.

50. Imagenet large scale visual recognition challenge / O. Russakovsky [et al.] // International journal of computer vision. — 2015. — Vol. 115, no. 3. — P. 211—252.

51. *Girshick, R.* Fast R-CNN / R. Girshick // Proceedings of the IEEE international conference on computer vision. — 2015. — P. 1440—1448.

52. Faster R-CNN: Towards real-time object detection with region proposal networks / S. Ren, K. He, R. Girshick, J. Sun // Advances in neural information processing systems. — 2015. — P. 91—99.

53. *Long, J.* Fully convolutional networks for semantic segmentation / J. Long, E. Shelhamer, T. Darrell // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2015. — P. 3431—3440.

54. Hedged deep tracking / Y. Qi [et al.] // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2016. — P. 4303—4311.

55. Mask R-CNN / K. He, G. Gkioxari, P. Dollár, R. Girshick // Proceedings of the IEEE international conference on computer vision. — 2017. — P. 2961—2969.

56. *Ramachandran, P.* Unsupervised Pretraining for Sequence to Sequence Learning / P. Ramachandran, P. Liu, Q. Le // Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. — Copenhagen, Denmark : Association for Computational Linguistics, 09/2017. — P. 383—391. — URL: https://www.aclweb.org/anthology/D17-1039.

57. Learned in Translation: Contextualized Word Vectors / B. McCann, J. Bradbury, C. Xiong, R. Socher // Advances in Neural Information Processing Systems 30 / ed. by I. Guyon [et al.]. — Curran Associates, Inc., 2017. — P. 6294—6305. — URL: http://papers.nips.cc/paper/7209-learned-in-translation-contextualized-word-vectors.pdf.

58. Semi-supervised sequence tagging with bidirectional language models / M. Peters, W. Ammar, C. Bhagavatula, R. Power // Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). — Vancouver, Canada : Association for Computational Linguistics, 07/2017. — P. 1756—1765. — URL: https://www.aclweb.org/anthology/P17-1161.

59. *Howard, J.* Universal Language Model Fine-tuning for Text Classification / J. Howard, S. Ruder // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). — Melbourne, Australia : Association for Computational Linguistics, 07/2018. — P. 328—339. — URL: https://www.aclweb.org/anthology/P18-1031.

60. Pre-trained models for natural language processing: A survey / X. Qiu [et al.] // arXiv preprint arXiv:2003.08271. — 2020.

61. XLNet: Generalized Autoregressive Pretraining for Language Understanding / Z. Yang [et al.] // Advances in Neural Information Processing Systems 32 / ed. by H. Wallach [et al.]. — Curran Associates, Inc., 2019. — P. 5753—5763. — URL: http://papers.nips.cc/paper/8812-xlnet-generalized-autoregressive-pretraining-for-language-understanding.pdf.

62. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context / Z. Dai [et al.] // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. — Florence, Italy : Association for Computational Linguistics, 07/2019. — P. 2978—2988. — URL: https://www.aclweb.org/anthology/P19-1285.

63. Roberta: A robustly optimized bert pretraining approach / Y. Liu [et al.] // arXiv preprint arXiv:1907.11692. — 2019.

64. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations / Z. Lan [et al.] // International Conference on Learning Representations. — 2020. — URL: https://openreview.net/forum?id=H1eA7AEtvS.

65. Universal Transformers / M. Dehghani [et al.] // International Conference on Learning Representations. — 2019. — URL: https://openreview.net/forum?id=HyzdRiR9Y7.

66. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension / M. Lewis [et al.] // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. — Online : Association for Computational Linguistics, 07/2020. — P. 7871—7880. — URL: https://www.aclweb.org/anthology/2020.acl-main.703.

67. Electra: Pre-training text encoders as discriminators rather than generators / K. Clark, M.-T. Luong, Q. V. Le, C. D. Manning // arXiv preprint arXiv:2003.10555. — 2020.

68. *Peters, M. E.* To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks / M. E. Peters, S. Ruder, N. A. Smith // Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019). — Florence, Italy : Association for Computational Linguistics, 08/2019. — P. 7—14. — URL: https://www.aclweb.org/anthology/W19-4302.

69. *Phang, J.* Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks / J. Phang, T. Févry, S. R. Bowman // arXiv preprint arXiv:1811.01088. — 2018.

70. A structured self-attentive sentence embedding / Z. Lin [et al.] // arXiv preprint arXiv:1703.03130. — 2017.

71. Deep residual learning for image recognition / K. He, X. Zhang, S. Ren, J. Sun // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2016. — P. 770—778.

72. *Ba, J. L.* Layer normalization / J. L. Ba, J. R. Kiros, G. E. Hinton // arXiv preprint arXiv:1607.06450. — 2016.

73. SQuAD: 100,000+ Questions for Machine Comprehension of Text / P. Rajpurkar, J. Zhang, K. Lopyrev, P. Liang // Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. — Austin, Texas : Association for Computational Linguistics, 11/2016. — P. 2383—2392. — URL: https://www.aclweb.org/anthology/D16-1264.

74. *Williams, A.* A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference / A. Williams, N. Nangia, S. Bowman // Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). — New Orleans, Louisiana : Association for Computational Linguistics, 2018. — P. 1112—1122. — URL: http://aclweb.org/anthology/N18-1101.

75. *Jernite, Y.* A fast variational approach for learning Markov random field language models / Y. Jernite, A. Rush, D. Sontag // International Conference on Machine Learning. — 2015. — P. 2209—2217.

76. *Wang, A.* Bert has a mouth, and it must speak: Bert as a markov random field language model / A. Wang, K. Cho // arXiv preprint arXiv:1902.04094. — 2019.

77. *Pires, T.* How Multilingual is Multilingual BERT? / T. Pires, E. Schlinger, D. Garrette // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. — Florence, Italy : Association for Computational Linguistics, 07/2019. — P. 4996—5001. — URL: https://www.aclweb.org/anthology/P19-1493.

78. *Wu, S.* Beto, Bentz, Becas: The Surprising Cross-Lingual Effectiveness of BERT / S. Wu, M. Dredze // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). — Hong Kong, China : Association for Computational Linguistics, 11/2019. — P. 833—844. — URL: https://www.aclweb.org/anthology/D19-1077.

79. *Artetxe, M.* On the Cross-lingual Transferability of Monolingual Representations / M. Artetxe, S. Ruder, D. Yogatama // Proceedings of the 58th Annual

Meeting of the Association for Computational Linguistics. — Online : Association for Computational Linguistics, 07/2020. — P. 4623—4637. — URL: https://www.aclweb.org/anthology/2020.acl-main.421.

80. Exporing the BERT Cross-Lingual Transfer for Reading Comprehension / V. Konovalov [et al.] // Computational Linguistics and Intellectual Technologies. International Conference" Dialogue 2020" Proceedings. — 2020. — P. 445—453.

81. *Shavrina, T.* TO THE METHODOLOGY OF CORPUS CONSTRUCTION FOR MACHINE LEARNING:"TAIGA" SYNTAX TREE CORPUS AND PARSER / T. Shavrina, O. Shapovalova // КОРПУСНАЯ ЛИНГВИСТИКА–2017. — 2017. — С. 78—84.

82. *Lison, P.* Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles / P. Lison, J. Tiedemann. — 2016.

83. DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset / Y. Li [et al.] // Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers). — 2017. — P. 986—995.

84. Conversational Flow in Oxford-style Debates / J. Zhang, R. Kumar, S. Ravi, C. Danescu-Niculescu-Mizil // Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. — 2016. — P. 136—141.

85. Effects of age and gender on blogging. / J. Schler, M. Koppel, S. Argamon, J. W. Pennebaker // AAAI spring symposium: Computational approaches to analyzing weblogs. Vol. 6. — 2006. — P. 199—205.

86. *Efimov, P.* SberQuAD–Russian Reading Comprehension Dataset: Description and Analysis / P. Efimov, L. Boytsov, P. Braslavski // arXiv preprint arXiv:1912.09723. — 2019.

87. Roberta: A robustly optimized bert pretraining approach / Y. Liu [et al.] // arXiv preprint arXiv:1907.11692. — 2019.

88. ParaPhraser: Russian paraphrase corpus and shared task / L. Pivovarova, E. Pronoza, E. Yagunova, A. Pronoza // Conference on Artificial Intelligence and Natural Language. — Springer. 2017. — P. 211—225.

89. RuSentiment: An Enriched Sentiment Analysis Dataset for Social Media in Russian / A. Rogers [et al.] // Proceedings of the 27th International Conference on Computational Linguistics. — 2018. — P. 755—763.

90. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank / R. Socher [et al.] // Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. — Seattle, Washington, USA : Association for Computational Linguistics, 2013. — P. 1631—1642. — URL: https://www.aclweb.org/anthology/D13-1170.

91. Neural Architectures for Named Entity Recognition / G. Lample [et al.] // Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. — San Diego, California : Association for Computational Linguistics, 06/2016. — P. 260—270. — URL: https://www.aclweb.org/anthology/N16-1030.

92. *Mozharova, V.* Two-stage approach in Russian named entity recognition / V. Mozharova, N. Loukachevitch // 2016 International FRUCT Conference on Intelligence, Social Media and Web (ISMW FRUCT). — 2016. — Sept. — P. 1—6.

93. *Власова, Н.* Сообщение о русскоязычной коллекции для задачи извлечения личных имен из текстов / Н. Власова, Е. Сулейманова, И. Трофимов // Труды конференции по компьютерной и когнитивной лингвистике TEL'2014 "Языковая семантика: модели и технологии". — Казань, Россия, 2014. — С. 36—40.

94. The Second Cross-Lingual Challenge on Recognition, Classification, Lemmatization, and Linking of Named Entities across Slavic Languages / J. Piskorski [et al.] // Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing. — Florence, Italy : Association for Computational Linguistics, 2019.

95. *Kravchenko, D.* Paraphrase detection using machine translation and textual similarity algorithms / D. Kravchenko // Conference on Artificial Intelligence and Natural Language. — Springer. 2017. — P. 277—292.

96. *Levesque, H.* The winograd schema challenge / H. Levesque, E. Davis, L. Morgenstern // Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning. — 2012.

97. *Cai, J.* Evaluation metrics for end-to-end coreference resolution systems / J. Cai, M. Strube // Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue. — Association for Computational Linguistics. 2010. — P. 28—36.

98. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes / S. Pradhan [et al.] // Joint Conference on EMNLP and CoNLL-Shared Task. — 2012. — P. 1—40.

99. A model-theoretic coreference scoring scheme / M. Vilain [et al.] // Proceedings of the 6th conference on Message understanding. — Association for Computational Linguistics. 1995. — P. 45—52.

100. *Bagga, A.* Algorithms for scoring coreference chains / A. Bagga, B. Baldwin // The first international conference on language resources and evaluation workshop on linguistics coreference. Vol. 1. — Granada, Spain. 1998. — P. 563—566.

101. *Luo, X.* On coreference resolution performance metrics / X. Luo // Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing. — Association for Computational Linguistics. 2005. — P. 25—32.

102. *Ng, V.* Machine learning for entity coreference resolution: A retrospective look at two decades of research / V. Ng // Thirty-First AAAI Conference on Artificial Intelligence. — 2017.

103. CoNLL-2011 Shared Task: Modeling Unrestricted Coreference in OntoNotes / S. Pradhan [et al.] // Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task. — Portland, Oregon, USA : Association for Computational Linguistics, 06/2011. — P. 1—27. — URL: https://www.aclweb.org/anthology/W11-1901.

104. *Fernandes, E.* Latent Structure Perceptron with Feature Induction for Unrestricted Coreference Resolution / E. Fernandes, C. dos Santos, R. Milidiú // Joint Conference on EMNLP and CoNLL - Shared Task. — Jeju Island, Korea : Association for Computational Linguistics, 07/2012. — P. 41—48. — URL: https://www.aclweb.org/anthology/W12-4502.

105. *Freund, Y.* Large margin classification using the perceptron algorithm / Y. Freund, R. E. Schapire // Machine learning. — 1999. — Vol. 37, no. 3. — P. 277—296.

106. Learning Anaphoricity and Antecedent Ranking Features for Coreference Resolution / S. Wiseman, A. M. Rush, S. Shieber, J. Weston // Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). — Beijing, China : Association for Computational Linguistics, 07/2015. — P. 1416—1426. — URL: https://www.aclweb.org/anthology/P15-1137.

107. Evaluating Anaphora and Coreference Resolution for Russian / S. Toldova [et al.] // Komp'juternaja lingvistika i intellektual'nye tehnologii. Po materialam ezhegodnoj Mezhdunarodnoj konferencii «Dialog». — 2014. — P. 681—695.

108. Error analysis for anaphora resolution in Russian: new challenging issues for anaphora resolution task in a morphologically rich language / S. Toldova [et al.] // Proceedings of the Workshop on Coreference Resolution Beyond OntoNotes (CORBON 2016). — 2016. — P. 74—83.

109. RU-EVAL-2019: Evaluating Anaphora And Coreference Resolutoin For Russian : tech. rep. / E. Budnikov [et al.]. — 2019. — URL: http://www.dialog-21.ru/media/4689/budnikovzverevamaximova2019evaluatinganaphoracoreferencer pdf.

110. Anaphora analysis based on ABBYY Compreno linguistic technologies / A. V. Bogdanov, S. S. Dzhumaev, D. A. Skorinkin, A. S. Starostin // Computational Linguistics and Intellectual Technologies. International Conference" Dialogue 2014" Proceedings. — 2014. — P. 89—101.

111. Anaphoric annotation and corpus-based anaphora resolution: An experiment / E. V. Protopopova [et al.]. — 2014.

112. *Cortes, C.* Support-vector networks / C. Cortes, V. Vapnik // Machine learning. — 1995. — Vol. 20, no. 3. — P. 273—297.

113. *Ionov, M.* The impact of morphology processing quality on automated anaphora resolution for Russian / M. Ionov, A. Kutuzov // Computational Linguistics and Intellectual Technologies. International Conference" Dialogue 2014" Proceedings. — 2014.

114. *Sysoev, A. A.* Coreference Resolution in Russian: State-of-the-art approaches application and evolvement. / A. A. Sysoev, I. A. Andrianov, K. A. Y. // Computational Linguistics and Intellectual Technologies. International Conference" Dialogue 2017" Proceedings. — 2017. — P. 317—338.

115. *Soon, W. M.* A machine learning approach to coreference resolution of noun phrases / W. M. Soon, H. T. Ng, D. C. Y. Lim // Computational linguistics. — 2001. — Vol. 27, no. 4. — P. 521—544.

116. *Ng, V.* Improving Machine Learning Approaches to Coreference Resolution / V. Ng, C. Cardie // Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. — Philadelphia, Pennsylvania, USA : Association for Computational Linguistics, 07/2002. — P. 104—111. — URL: https://www.aclweb.org/anthology/P02-1014.

117. End-to-end Neural Coreference Resolution / K. Lee, L. He, M. Lewis, L. Zettlemoyer // Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. — 2017. — P. 188—197.

118. *Lee, K.* Higher-Order Coreference Resolution with Coarse-to-Fine Inference / K. Lee, L. He, L. Zettlemoyer // Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers). — New Orleans, Louisiana : Association for Computational Linguistics, 06/2018. — P. 687—692. — URL: https://www.aclweb.org/anthology/N18-2108.

119. *Toldova, S.* Coreference Resolution for Russian: The Impact of Semantic Features / S. Toldova, I. Maxim // Computational Linguistics and Intellectual Technologies. International Conference" Dialogue 2017" Proceedings. — 2017. — P. 339—349.

120. *Inshakova, E.* An anaphora resolution system for Russian based on ETAP-4 linguistic processor / E. Inshakova // Komp'yuternaya lingvistika i intellektual'nye tekhnologii. Po materialam ezhegodnoi Mezhdunarodnoi konferentsii'Dialog. — 2019. — P. 239—251.

121. BERT for Coreference Resolution: Baselines and Analysis / M. Joshi, O. Levy, L. Zettlemoyer, D. Weld // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). — Hong Kong, China : Association for Computational Linguistics, 11/2019. — P. 5803—5808. — URL: https://www.aclweb.org/anthology/D19-1588.

122. SpanBERT: Improving Pre-training by Representing and Predicting Spans / M. Joshi [et al.] // Transactions of the Association for Computational Linguistics. — 2020. — Vol. 8. — P. 64—77. — URL: https://transacl.org/ojs/index.php/tacl/article/view/1853.

123. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia / J. Lehmann [et al.] // Semantic web. — 2015. — Vol. 6, no. 2. — P. 167—195.

124. *Vrandečić, D.* Wikidata: a free collaborative knowledgebase / D. Vrandečić, M. Krötzsch // Communications of the ACM. — 2014. — Vol. 57, no. 10. — P. 78—85.

125. Freebase: a collaboratively created graph database for structuring human knowledge / K. Bollacker [et al.] // Proceedings of the 2008 ACM SIGMOD international conference on Management of data. — 2008. — P. 1247—1250.

126. *Wang, S.* Machine Comprehension Using Match-LSTM and Answer Pointer / S. Wang, J. Jiang // 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. — OpenReview.net, 2017. — URL: https://openreview.net/forum?id=B1-q5Pqxl.

127. Bidirectional Attention Flow for Machine Comprehension / M. J. Seo, A. Kembhavi, A. Farhadi, H. Hajishirzi // 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. — OpenReview.net, 2017. — URL: https://openreview.net/forum?id=HJ0UKP9ge.

128. Reinforced mnemonic reader for machine reading comprehension / M. Hu [et al.] // Proceedings of the 27th International Joint Conference on Artificial Intelligence. — AAAI Press. 2018. — P. 4099—4106.

129. Ms marco: A human generated machine reading comprehension dataset / P. Bajaj [et al.] // arXiv preprint arXiv:1611.09268. — 2016.

130. Reading Wikipedia to Answer Open-Domain Questions / D. Chen, A. Fisch, J. Weston, A. Bordes // Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). — Vancouver, Canada : Association for Computational Linguistics, 07/2017. — P. 1870—1879. — URL: https://www.aclweb.org/anthology/P17-1171.

131. *Clark, C.* Simple and Effective Multi-Paragraph Reading Comprehension / C. Clark, M. Gardner // Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). — Melbourne, Australia : Association for Computational Linguistics, 07/2018. — P. 845—855. — URL: https://www.aclweb.org/anthology/P18-1078.

132. *Vinyals, O.* Pointer Networks / O. Vinyals, M. Fortunato, N. Jaitly // Advances in Neural Information Processing Systems 28 / ed. by C. Cortes [et al.]. — Curran Associates, Inc., 2015. — P. 2692—2700. — URL: http://papers.nips.cc/paper/5866-pointer-networks.pdf.

133. *Provilkov, I.* BPE-Dropout: Simple and Effective Subword Regularization / I. Provilkov, D. Emelianenko, E. Voita // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. — Online : Association for Computational Linguistics, 07/2020. — P. 1882—1892. — URL: https://www.aclweb.org/anthology/2020.acl-main.170.

134. *Bostrom, K.* Byte pair encoding is suboptimal for language model pretraining / K. Bostrom, G. Durrett // arXiv preprint arXiv:2004.03720. — 2020.

135. *Hinton, G.* Distilling the knowledge in a neural network / G. Hinton, O. Vinyals, J. Dean // arXiv preprint arXiv:1503.02531. — 2015.

Список рисунков

Список таблиц