ЛЕКЦИЯ 2. Трансформеры

Обзор архитектуры трансформеров

Модели-трансформеры вызвали огромный интерес из-за их эффективности в огромном диапазоне задач NLP, от классификации до генерации текста. Критически важной частью этих моделей является механизм внимания. Впрочем, еще до появления трансформеров механизм внимания начали применять для улучшения традиционных моделей глубокого обучения, таких как RNN.

Механизм внимания

Один из первых вариантов механизма внимания был предложен Дмитрием Богдановым в 2015 г. Этот механизм основан на том факте, что модели на основе RNN (рекуррентные нейронные сети), такие как GRU (управляемые рекуррентные блоки) или LSTM (долгая краткосрочная память), имеют информационное узкое место для таких задач, как нейронный машинный перевод (neural machine translation, NMT). Эти модели на основе энкодера-декодера получают ввод в виде token-id и подвергают его рекуррентной обработке (энкодер). После этого обработанное промежуточное представление подается в другой рекуррентный блок (декодер) для извлечения результатов. Эта лавинообразная информация подобна катящемуся снежному шару, на который налипает вся информация, и «раскатать» этот шар обратно – крайне сложная задача, потому что декодирующая часть не видит всех зависимостей и получает в качестве входных данных только промежуточное представление (вектор контекста).

Чтобы устранить это узкое место и согласовать декодер с энкодером, Богданов предложил механизм внимания, в котором используются веса, присвоенные промежуточным скрытым значениям. Эти веса определяют количество внимания, которое модель должна уделять входным данным на каждом этапе декодирования. Наличие подобных замечательных подсказок очень помогает моделям в таких задачах, как NMT (нейронный машинный перевод), которые относятся к задачам типа «многие ко многим». На рис. 1 представлена схема типичного механизма внимания.

Впоследствии были предложены разные механизмы внимания с разнообразными улучшениями. В семейство этих механизмов входят аддитивное, мультипликативное, общее внимание, а также внимание на основе скалярного произведения (dot-product attention, DPA). Модифицированная версия последнего механизма, имеющая весовой параметр, известна как взвешенное скалярное произведение (scaled dot-product). Этот особый тип внимания лежит в основе моделей трансформеров. Мы будем с определенной долей условности называть его механизмом многопоточного внимания (multi-head attention mechanism).

Механизм аддитивного внимания также оказал заметное влияние на задачи NMT (нейронный машинный перевод).

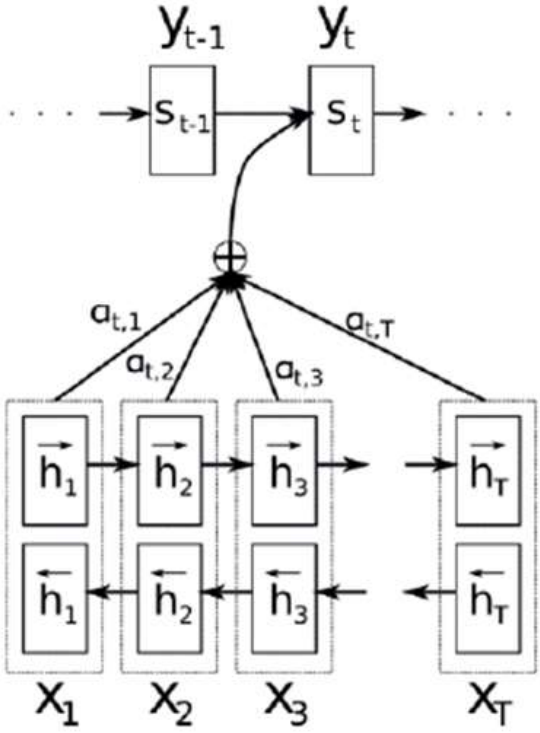

Рис. 1. Схема типичного механизма внимания

В табл. 1 представлен обзор различных типов механизмов внимания.

Таблица 1. Типы механизмов внимания

Название	Функция внимания
Контентное внимание (content-based attention)	$score(s_i, h_i) = cos[s_i, h_i]$
Аддитивное (additive)	$score(s_i, h_i) = \mathrm{v}_\mathrm{a}^\intercal tanh(\mathrm{W}_\mathrm{a}[s_t, h_i])$
Позиционное (location-based)	$\alpha_{t,i} = \mathrm{softmax}(\mathrm{W}_\mathrm{a} s_t)$
Общее (general)	$score(s_i, h_i) = s_t^\intercal \mathrm{W}_\mathrm{a} h_i$
Скалярное произведение (dot-product)	$score(s_i, h_i) = s_t^\intercal h_i$
Взвешенное скалярное произведение (scaled dot product)	$score(s_i, h_i) = \frac{s_t^\intercal h_i}{\sqrt{n}}$

Поскольку механизмы внимания не являются специфическими для NLP, их также применяют в различных областях и сценариях, от компьютерного зрения до распознавания речи. На рис. 2 показана визуализация процесса мультимодального обучения нейросети созданию подписей к изображениям.

Рис. 2. Механизм внимания в компьютерном зрении

Механизм многопоточного внимания, схематически представленный на рис.3, является важной частью архитектуры трансформеров:

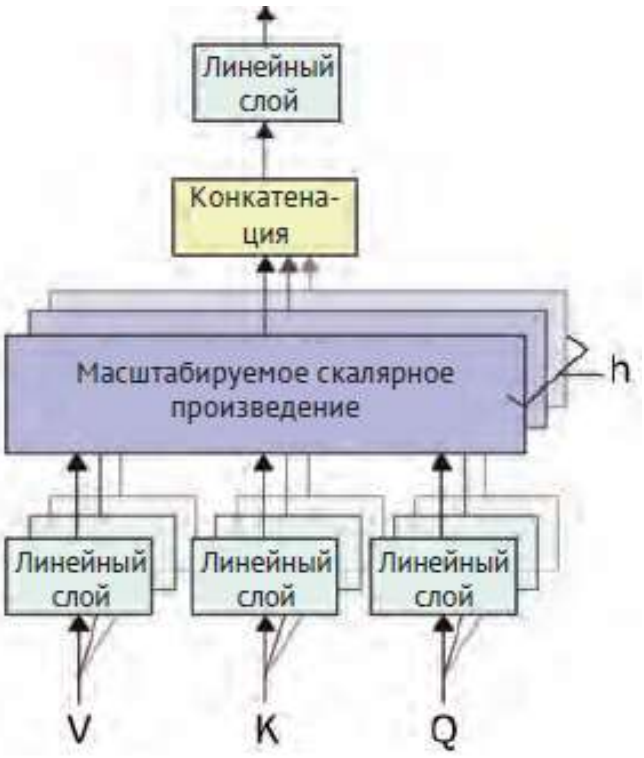

Рис. 3. Механизм многопоточного внимания

Механизмы многопоточного внимания

Прежде чем переходить к механизмам внимания на основе взвешенных скалярных произведений, необходимо разобраться в том, что такое самовнимание

(self-attention). Механизм самовнимания, схематически представленный на рис. 4, является базовой формой механизма, взвешенного самовнимания

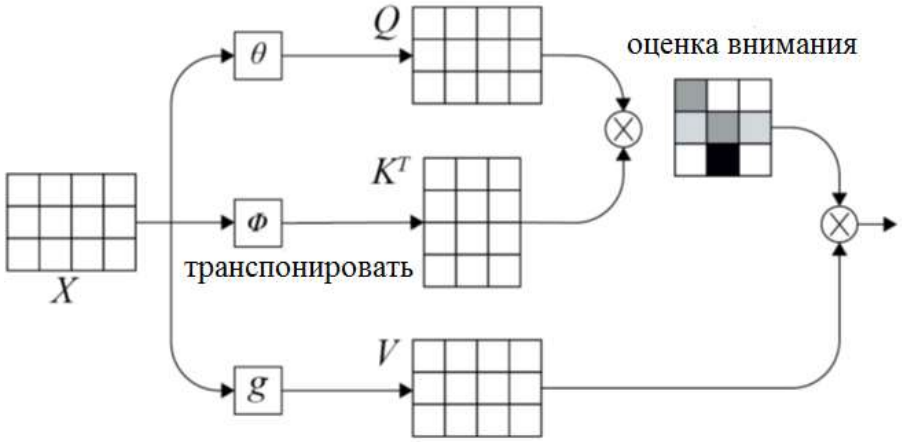

Рис. 4. Математическое представление механизма внимания

Этот механизм получает матрицу ввода X и производит оценку внимания ее элементов. В нашем случае X представляет собой матрицу 3×4, где 3 – количество токенов, а 4 – размер представления. Матрица Q на рис. 4 называется запросом (query), K – ключом (key), а V – значением (value). Эти матрицы получаются в результате перемножения X и матриц, помеченных символами Θ, Φ и g соответственно. Перемножение матриц запроса (Q) и ключа (K) дает матрицу весов внимания. Этот механизм также можно рассматривать как базу данных, в которой мы используем запрос и ключи, чтобы получить численную оценку связей между различными элементами. Перемножение весов внимания и матрицы V дает конечный результат механизма внимания этого типа. Основная причина, по которой он называется самовниманием, – это его единственный вход X; матрицы Q, K и V вычисляются из X.

Механизм взвешенного DPA (dot-product attention – внимание на основе скалярного произведения) очень похож на механизм самовнимания (скалярного произведения), за исключением того, что он использует весовой коэффициент. С другой стороны, многопоточность гарантирует, что модель способна рассматривать различные аспекты ввода на всех уровнях. Модели-трансформеры учитывают аннотации энкодера и скрытые значения из прошлых слоев. Архитектура трансформера не имеет рекуррентного пошагового потока; вместо этого в ней применяется позиционное кодирование, чтобы иметь информацию о позиции каждого токена во входной последовательности. На слои в первой части энкодера подаются входные данные, представляющие собой объединенные значения представлений (инициализированные случайным образом) и фиксированные значения позиционного кодирования; они распространяются по архитектуре, как показано на рис. 5.

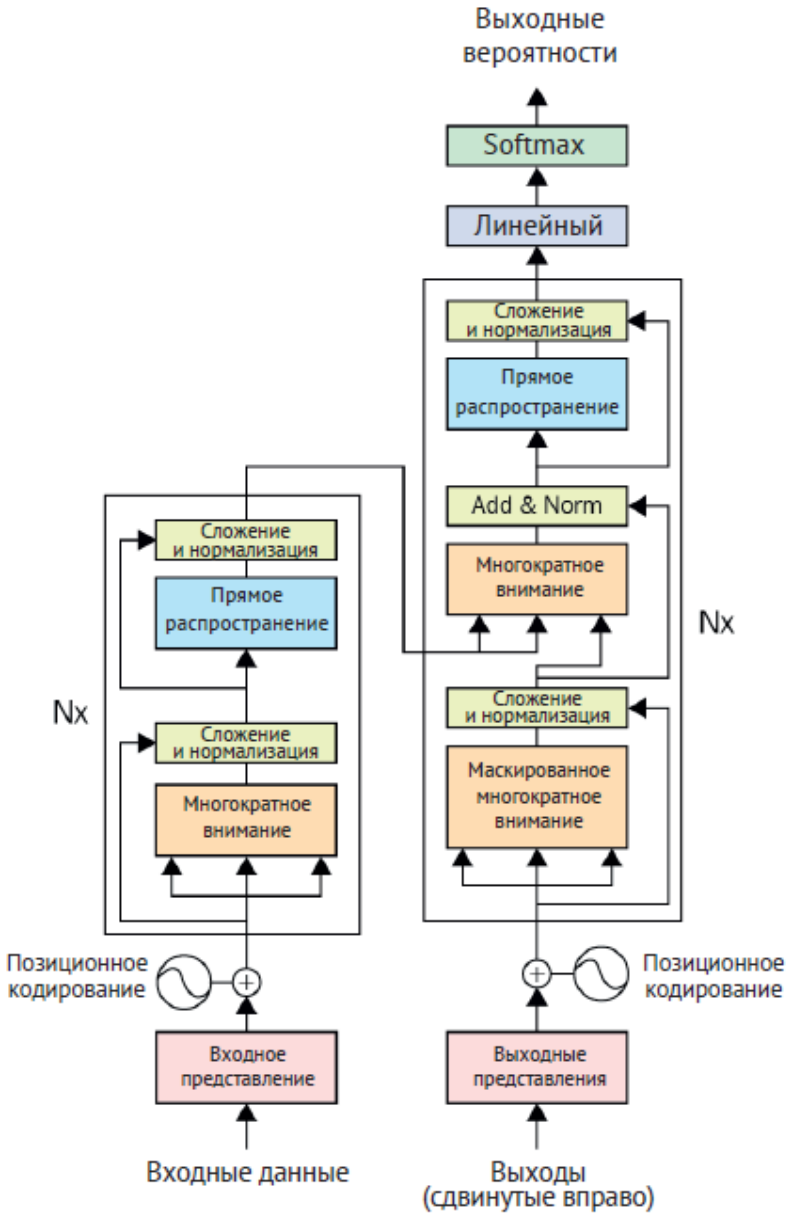

Рис. 5. Обобщенная схема трансформера

Информация о положении получается путем вычисления синусоидальных и косинусоидальных волн на разных частотах. Пример позиционного кодирования представлен на рис. 6.

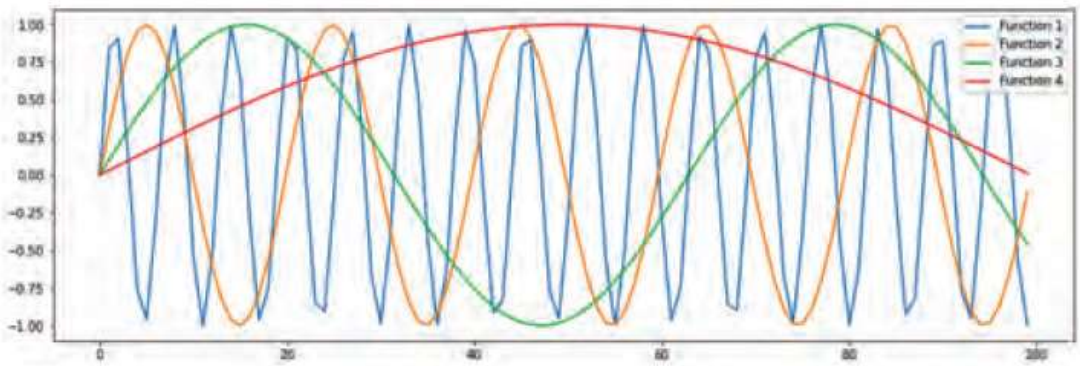

Рис. 6. Позиционное кодирование

Хороший пример качества работы трансформера и механизма взвешенного DPA приведен на широко известной схеме (рис. 7.), которая встречается во многих публикациях.

Рис. 7. Пример работы механизма внимания трансформера

Как видно на рис. 1.18, слово it в разных контекстах обозначает разные объ-

екты. (Например, в предложении Курица не перешла дорогу, потому что она была слишком испугана слово она связано со словом курица. В предложении Курица не перешла дорогу, потому что она была слишком широкая слово она связано со словом дорога). Еще одно улучшение, которое стало возможным благодаря трансформерам, – это параллелизм. Обычные последовательные рекуррентные модели, такие как LSTM и GRU, не имеют подобных возможностей, потому что они обрабатывают входные данные токен за токеном. С другой стороны, слои прямого распространения ускоряются немного больше, потому что умножение одной матрицы происходит значительно быстрее, чем рекуррентная операция. Стеки слоев многопоточного внимания позволяют лучше понимать сложные предложения. Хороший наглядный пример механизма многопоточного внимания представлен на рис. 8.

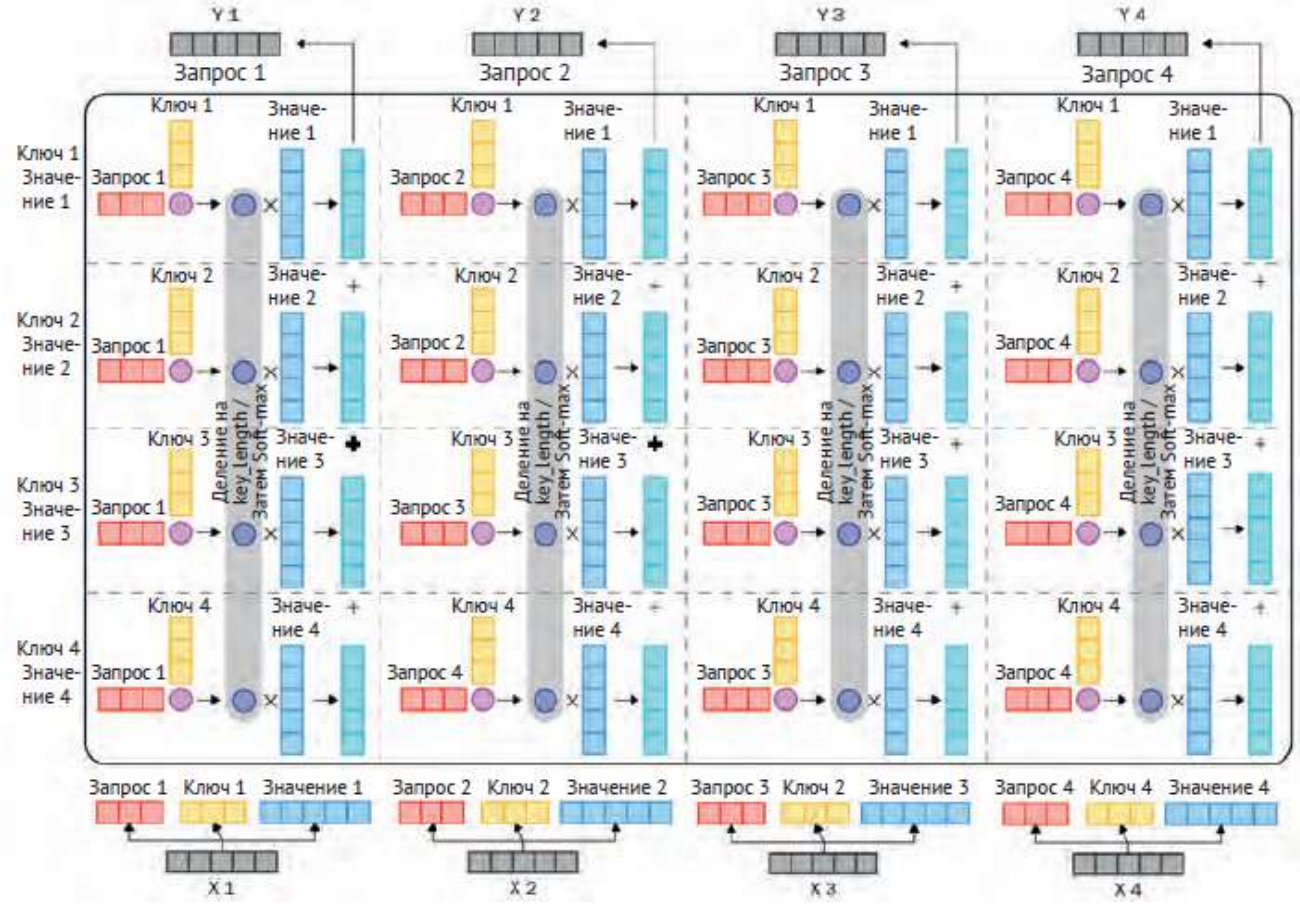

Рис. 8. Механизм многопоточного внимания

На стороне декодера механизма внимания используется подход, очень похожий на энкодер, но с небольшими изменениями. Механизм многопоточного внимания устроен аналогично, но в нем также используется выход стека энкодера, откуда данные передаются каждому стеку декодера на втором уровне механизма многопоточного внимания. Эта небольшая модификация открывает модели доступ к стеку энкодера в процессе декодирования и в то же время помогает ей во время обучения поддерживать лучший градиент потока через слои. Последний слой Softmax за слоем декодера предоставляет выходные данные для различных задач, таких как NMT (нейронный машинный перевод) для которых и была разработана оригинальная архитектура трансформеров.

Эта архитектура имеет два канала, обозначенных как входы и выходы (сдвинутые вправо). Один из них всегда присутствует (входы) как при обучении, так и при выводе, в то время как другой присутствует только при обучении и выводе, который производится моделью. Причина, по которой мы не используем прогнозы модели для вывода, заключается в том, чтобы не позволить модели зайти слишком далеко по неправильному пути. Но что это значит? Представьте себе нейронную модель перевода, которая пытается перевести предложение с английского на французский – на каждом шаге она предсказывает слово и

использует его для предсказания следующего слова. Но если на каком-то этапе что-то пойдет не так, все последующие прогнозы тоже окажутся неверными. Чтобы модель не пошла по неправильному пути, мы предоставляем правильные слова в виде версии со сдвигом вправо.

Наглядный пример модели трансформера приведен на рис. 9. Здесь показана модель трансформера с двумя энкодерами и двумя слоями декодера. Слой «Сложение и нормализация» на этой схеме прибавляет и нормализует ввод, который он принимает от слоя «Прямое распространение»:

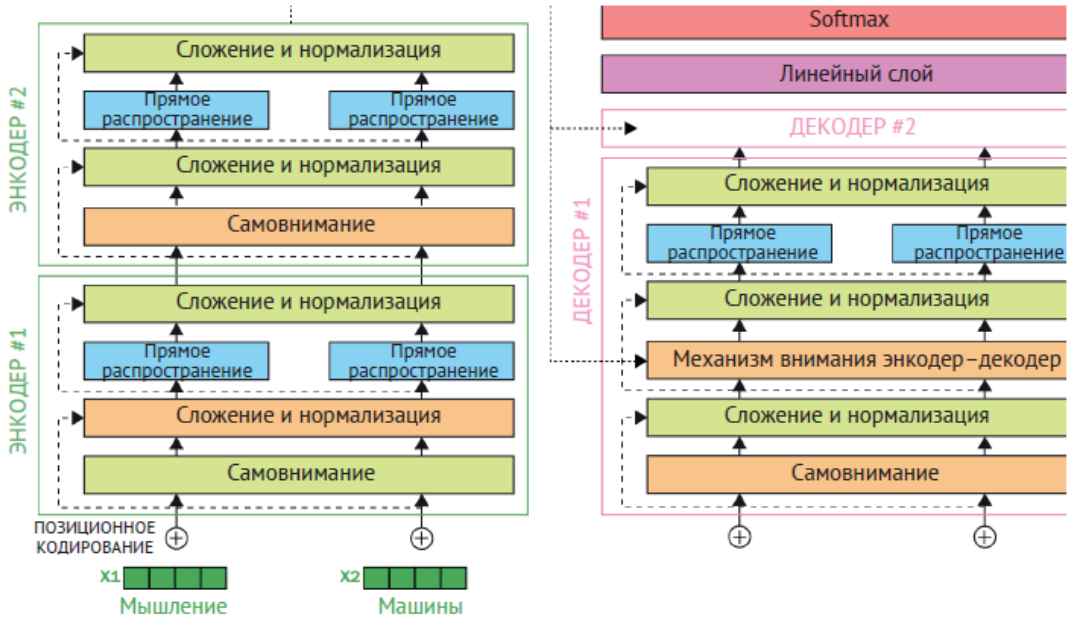

Рис. 9. Модель трансформера

Другое важное усовершенствование, которое используется в архитектуре на основе трансформеров, основано на простой универсальной схеме сжатия текста для предотвращения появления невидимых токенов на стороне ввода. Этот подход, который осуществляется с использованием различных методов, таких как кодирование пар байтов и кодирование частей предложения, улучшает качество работы трансформеров с невидимыми токенами. Он также помогает модели, когда ей встречаются морфологически близкие токены. Такие токены раньше никогда не попадались и редко применялись при обучении, но тем не менее их можно использовать для вывода. В некоторых случаях при обучении встречаются их фрагменты; последнее происходит в случае морфологически богатых языков, таких как турецкий, немецкий, чешский и латышский. Например, модель знает слово домик, но не домики.В таких случаях она может токенизировать слово домики как домик+и. Обе эти части модели известны.

Модели на основе трансформеров имеют много общего – например, все они основаны на общей оригинальной архитектуре, и разница лишь в том, какие шаги они используют, а какие не используют. В некоторых случаях вносятся

незначительные изменения, например улучшения в механизме многопоточного внимания.

Трансформеры и перенос обучения

Перенос обучения (transfer learning) – это область искусственного интеллекта и машинного обучения, цель которой – сделать модели повторно используемыми для различных задач. Например, модель, обученная для определенной задачи A, может быть повторно использована (тонко настроена) для другой задачи B. В области NLP этого добиваются с помощью таких архитектур, как трансформеры, которые могут сформировать понимание языка как такового, с помощью языкового моделирования. Такие модели называются языковыми моделями (language model) – они предоставляют модель языка, на котором их обучали. Перенос обучения – это не новый подход, он активно применяется в различных областях, таких как компьютерное зрение. В качестве примеров предварительно обученных моделей можно назвать ResNet, Inception, VGG и EfficientNet, которые можно настраивать для различных задач компьютерного зрения.

В области NLP также возможно неглубокое (shallow learning) обучение с переносом с использованием таких моделей, как Word2vec, GloVe и Doc2vec. Такое обучение называется неглубоким, потому что в основе этого типа обучения нет модели, а вместо нее используются предварительно обученные векторы для слов/токенов. Вы можете использовать эти модели представления токенов или документов, за которыми следует классификатор, или применять их в сочетании с другими моделями, такими как RNN, вместо использования случайных представлений.

Перенос обучения в NLP с использованием трансформеров также возможен, потому что эти модели способны изучать сам язык без каких-либо размеченных данных. Моделирование языка – это задача получения переносимых весов для решения различных задач. Масочное моделирование языка (masked language modeling) – это один из методов, используемых для изучения самого языка. Как и в случае с оконной моделью Word2vec для прогнозирования центральных токенов, при масочном моделировании языка применяется аналогичный подход с некоторыми важными отличиями. При заданной вероятности каждое слово маскируется и заменяется специальным токеном, например [MASK]. Языковая модель (в нашем случае модель на основе трансформера) должна предсказывать замаскированные слова. В отличие от Word2vec, где используется окно, на вход модели подается целое предложение, и на выходе должно быть то же предложение с восстановленными замаскированными словами.

Одной из первых моделей, в которых для языкового моделирования использовалась архитектура трансформеров, является BERT, основанная на энкодерной части трансформера. Масочное моделирование языка выполняется BERT с использованием того же метода, который был описан до и после обучения языковой модели. BERT – это переносимая языковая модель для различных задач в области NLP, таких как классификация токенов, классификация последовательностей или даже ответы на вопросы.

Каждая из упомянутых задач относится к задачам тонкой настройки BERT после обучения языковой модели. Модель BERT наиболее известна своими ключевыми характеристиками базовой модели трансформерного энкодера, и, изменяя эти характеристики, можно получить разные версии – малые, облегченные, базовые, большие и сверхбольшие. Контекстное представление позволяет модели извлекать правильное значение каждого слова в зависимости от контекста, в котором оно дано, – например, слово холодный может иметь разные значения в двух разных предложениях: холодный взгляд и холодный ветер. Ключевыми характеристиками модели являются количество слоев энкодерной части, входная и выходная размерности представления и количество механизмов многопоточного внимания, как показано на рис. 10.

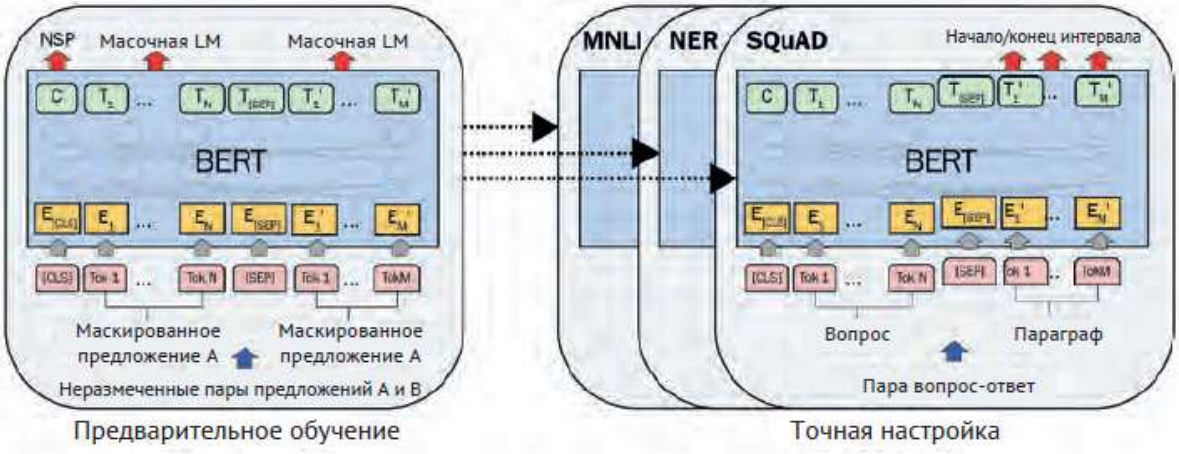

Рис. 10. Процедуры предварительного обучения и тонкой настройки BERT

Как показано на рис. 10, предварительная фаза обучения также содержит другую цель, известную как прогнозирование следующего предложения (nextsentence prediction). Как мы знаем, каждый документ состоит из предложений, следующих друг за другом, и еще одна важная часть обучения модели пониманию языка заключается в понимании отношений между предложениями – другими словами, связаны они между собой или нет. Для решения этих задач в BERT имеются специальные токены, такие как [CLS] и [SEP]. Не имеющий изначального смысла токен [CLS] применяется в качестве стартового токена для всех задач и содержит всю информацию о предложении. В

задачах классификации последовательности, таких как NSP, используется классификатор поверх выходных данных этого токена (выходная позиция 0). Он также полезен для оценки смысла предложения или получения его семантики, например при использовании сиамской модели BERT очень полезно сравнить два токена [CLS] для разных предложений по такому критерию, как косинусное подобие. В свою очередь, токен [SEP] используется только для различения двух предложений. Если после предварительного обучения кто-то захочет настроить BERT для задачи классификации последовательностей, такой как анализ тональности, ему следует использовать классификатор поверх выходного представления [CLS]. Также примечательно, что все модели с переносом обучения можно «заморозить» при тонкой настройке или, наоборот, «разморозить». Замораживание означает остановку обучения, после чего мы можем видеть все веса и смещения внутри модели как константы. В случае анализа тональности будет обучаться только классификатор, а не модель, если она заморожена.

Знакомство с трансформерами на практике

Для реализации трансформера необходимо установить библиотеки и программы, перечисленные ниже. Хотя наличие последней версии программы – это несомненный плюс, необходимо обязательно устанавливать версии, совместимые друг с другом.

- Анаконда;
- Transformer 4.0.0;
- PyTorch 1.1.0;
- TensorFlow 2.4.0;
- Datasets 1.4.1.

С установками этих библиотек Вы наверняка уже знакомы. Однако имеет смысл поговорить о важных инструментах сравнительного тестирования (бенчмарках), таких как General Language Understanding Evalution (GLUE – Общая оценка понимания языка), Cross-lingual TRansfer Evaluation of Multilingual Encoders (XTREME – Межъязыковая оценка трансферта многоязычных кодировщиков) и Stanford Question Answering Dataset (SquAD – Стэнфордский набор данных для ответов на вопросы). Сравнительное тестирование особенно важно при переносе обучения в многозадачных и многоязычных средах. В NLP мы в основном сосредоточены на конкретной метрике, которая представляет собой оценку качества решения определенной задачи или набора данных. Благодаря библиотеке Transformer мы можем перенести знания, выученные при решении конкретной задачи, в связанную задачу – этот процесс называется

переносом обучения. Перенося представления между связанными задачами, мы можем обучать модели общего назначения, которые предоставляют общие лингвистические знания разным задачам. Этот подход также известен как многозадачное обучение (multi-task learning, MTL). Другой аспект TL (transfer learning – перенос обучения) – передача знаний через естественные языки (многоязычные модели).

Бенчмарки

Рассмотрим следующие бенчмарки

GLUE

Недавние исследования опирались на тот факт, что подход многозадачного обучения позволяет достигать лучших результатов, чем однозадачное обучение, т.е. обучение отдельной модели для конкретной задачи. Исходя из этой предпосылки, и был разработан бенчмарк GLUE для MTL (многозадачное обучение), который представляет собой инструменты и наборы данных для оценки производительности моделей MTL по целому списку задач. Он предлагает общедоступную таблицу лидеров для мониторинга производительности, а также однозначную метрику, обобщающую 11 задач. Этот бенчмарк содержит множество заданий на понимание предложений, основанных на реальных задачах, охватывающих различные наборы данных разного размера, разные типы текста и уровни сложности. Задачи подразделяются на три следующих типа:

- задания, состоящие из одного предложения.

CoLA (Corpus of Linguistic Acceptability – Корпус лингвистической приемлемости) – набор данных о лингвистической приемлемости. Эта задача состоит из суждений о приемлемости в английском языке, взятых из статей по теории лингвистики;

SST-2 (набор данных Stanford Sentiment Treebank – Стэнфордское дерево настроений) – это задание включает в себя предложения из обзоров фильмов и человеческие аннотации их тональности с метками pos/neg (позитивная/негативная);

- задачи схожести и перефразирования.

MRPC (набор данных Microsoft Research Paraphrase Corpus – Корпус парафразов исследований Microsoft) – это задача на определение, являются ли предложения в паре семантически эквивалентными;

QQP (набор данных Quora Question Pairs – Пары вопросов Quora) – это задача на определение, является ли пара вопросов семантически эквивалентной;

STS-B (набор данных Semantic Textual Similarity Benchmark – Тест семантического текстового сходства). Это задание представляет собой набор пар предложений, взятых из заголовков новостей, с оценкой сходства от 1 до 5;

- задачи вывода

MNLI (Multi-Genre Natural Language Inference - Многожанровый корпус логических выводов на естественном языке) Это набор пар предложений с текстовым следствием. Задача состоит в том, чтобы предсказать, содержит ли текст гипотезу (следствие), противоречит гипотезе (противоречие) или не противоречит (нейтральный);

QNLI (набор данных Question Natural Language Inference – Вопрос Вывод на естественном языке) – это модифицированная версия SquAD. Задача – проверить, есть ли в предложении ответ на вопрос;

RTE (набор данных Recognizing Textual Entailment – Распознавание текстовых последствий) – это задача текстового вывода, связанная с объединением данных из различных источников. Этот набор данных похож на предыдущий набор данных QNLI, где задача состоит в том, чтобы проверить, следует ли за первым текстом второй;

WNLI (задача Winograd Natural Language Inference – Вывод Winograd (видимо фамилия Виноград) на естественном языке) – изначально это задача разрешения местоимения, связывающая местоимение и фразу в предложении. В GLUE эта задача преобразована в классификацию пар предложений, как будет описано далее.

SuperGLUE

Как и GLUE, SuperGLUE – это новый бенчмарк, созданный с обновленным набором более сложных задач понимания языка и предлагающий на данный момент общедоступный рейтинг из примерно восьми языковых задач, основанных на существующих данных, связанных с однозначной метрикой производительности, такой как у GLUE. SuperGLUE предлагает более сложную и разнообразную задачу в области универсальных технологий понимания языков.

XTREME

В последние годы исследователи, работающие в области NLP, все больше склоняются к изучению универсальных представлений, а не одной задачи, которую можно применить к другим связанным задачам. Иной способ построения языковой модели общего назначения – использование многоязычных задач. Было отмечено, что недавно разработанные многоязычные модели, такие как Multilingual BERT (mBERT) и XLM-R, предварительно обученные на огромном количестве многоязычных корпусов, показали лучшие результаты при их

переносе на другие языки. Таким образом, основным преимуществом здесь является то, что кросс-языковая генерализация позволяет нам создавать успешные приложения NLP на языках с ограниченными ресурсами при помощи межъязыкового переноса без подготовки.

Для работы в этом направлении был разработан бенчмарк XTREME. В настоящее время он включает около 40 различных языков, принадлежащих к 12 языковым семействам, и 9 различных задач, требующих логического вывода для различных уровней синтаксиса или семантики. Однако по-прежнему сложно масштабировать модель, чтобы охватить более 7000 языков мира, и приходится искать компромисс между языковым охватом и возможностями модели

XGLUE

XGLUE – еще один кросс-языковой тест для оценки и улучшения производительности кросс-языковых предварительно обученных моделей для понимания естественного языка (NLU) и генерации текстов на естественном языке (NLG). Первоначально он состоял из 11 заданий на 19 языках. Основное отличие от XTREME заключается в том, что обучающие данные для каждой задачи доступны только на английском языке. Это заставляет языковые модели учиться лишь на основе текстовых данных на английском языке и передавать эти знания на другие языки – это называется возможностью межъязыкового переноса без подготовки (cross-language zero-shot transfer). Второе отличие в том, что в XGLUE есть задачи одновременно для NLU и NLG.

SQuAD

Он предоставляет из себя набор пар «вопрос–ответ» для тестирования возможностей понимания прочитанного моделями NLP. Он состоит из списка вопросов, блоков текста для чтения и ответов, размеченных волонтерами на основе статей в Википедии. Ответом на вопрос является отрывок текста из блока для чтения. Первоначальная версия, SQuAD1.1, не предусматривает вариант вопроса без ответа, когда просто происходит сбор данных, поэтому на каждый вопрос есть ответ, который можно найти в каком-нибудь блоке для чтения. Модель NLP вынуждена дать ответ на вопрос, даже если это кажется невозможным. SQuAD2.0 – это улучшенная версия, в соответствии с которой модели NLP должны не только отвечать на вопросы, когда это возможно, но также должны воздерживаться от ответа, когда на вопрос невозможно ответить. SQuAD2.0 содержит 50 000 вопросов, на которые невозможно ответить, коварно разработанных волонтерами таким образом, чтобы они были похожи на вопросы, имеющие ответ. Кроме того, SQuAD2.0 также содержит 100 000 вопросов, взятых из SQuAD1.1.