ЛЕКЦИЯ 4. Авторегрессивные языковые модели

Работа с языковыми моделями AR (autoregressive – авторегрессивных языковых моделей)

Первоначально архитектура трансформеров была предназначена для решения задач Seq2Seq, таких как машинный перевод или резюмирование, но теперь она широко применяется в различных задачах NLP, начиная от классификации токенов и заканчивая разрешением кореферентности7 (coreference).

Кореферéнтность, или референциональное тождество – отношение между именами – компонентами высказывания, в котором имена ссылаются на один и тот же объект внеязыковой действительности. Например, имя средство для чистки кореферентно имени чистящее средство

Впоследствии в моделях начали использовать по отдельности и более творчески левую и правую части архитектуры. Общая цель, также известная как цель шумоподавления (denoising objective), заключается в том, чтобы полностью восстановить исходный вход из поврежденного входа в двунаправленном режиме, как показано в левой части рис. 4.1.

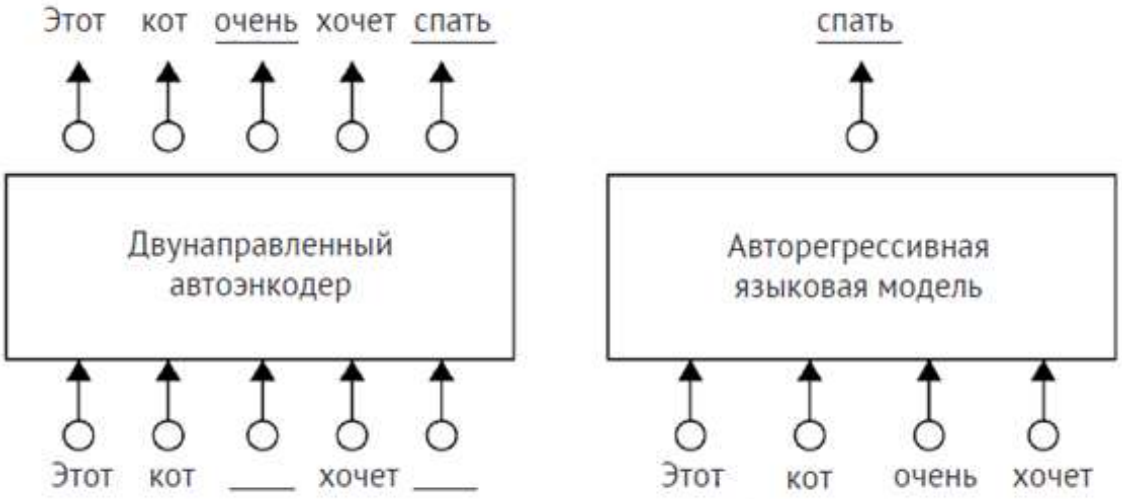

Рис. 4.1. Языковые модели AE и AR

Как видно из архитектуры BERT, которая является ярким примером автоэнкодерных моделей, они могут включать контекст, окружающий слово с обеих сторон. Однако первая проблема заключается в том, что искажающие символы [MASK], которые используются на этапе предварительного обучения, отсутствуют в данных на этапе тонкой настройки, что приводит к несоответствию между предварительным обучением и тонкой настройкой. Во-вторых, модель BERT небезосновательно предполагает, что замаскированные токены независимы друг от друга.

В свою очередь, авторегрессивные модели избегают таких предположений относительно независимости и, естественно, не страдают от несоответствия предварительного обучения, поскольку они полагаются на цель предсказания следующего токена, обусловленного предыдущими токенами без маскировки. Они просто используют декодирующую часть трансформера с маскированным самовниманием. Они предотвращают доступ модели к словам справа от текущего слова в прямом направлении (или слева от текущего слова в обратном направлении), что называется однонаправленностью. Их также называют каузальными языковыми моделями (causal language models, CLM) из-за их однонаправленности. На рис. 4.1 показана разница между моделями AE и AR.

Модель GPT и две построенные на ее основе модели (GPT-2, GPT-3) – Transformer-XL и XLNet – являются одними из наиболее упоминаемых в литературе популярных моделей AR. Несмотря на то что модель XLNet основана на авторегрессии, ей каким-то образом удается использовать обе контекстные стороны слова двунаправленным образом с помощью языковой цели, основанной на перестановках.

Устройство моделей на основе GPT

Модели AR состоят из нескольких блоков трансформеров. Каждый блок содержит маскированный слой многопоточного самовнимания вместе с точечным слоем прямого распространения. Активация в последнем блоке трансформера передается в функцию Softmax, которая генерирует распределения вероятностей слов по всему словарю слов для предсказания следующего слова.

Архитектура GPT (Improving Language Understanding by Generative Pre-Training (улучшение понимания языка с помощью генеративной предварительной подготовки), 2018), имеет несколько узких мест, которым подвержены традиционные конвейеры обработки естественного языка на основе машинного обучения. Например, эти конвейеры требуют, как огромного количества данных для конкретных задач, так и подгонки архитектуры специально под конкретные задачи. Кроме того, трудно применить входные преобразования для решения определенной задачи с минимальными изменениями в архитектуре предварительно обученной модели. Исходная архитектура GPT и ее преемники (GPT-2 и GPT-3), разработанные командой OpenAI, были сосредоточены на устранении этих узких мест. Главный вклад оригинального исследования авторов GPT состоит в том, что предварительно обученная модель дала удовлетворительные результаты не только для одной задачи, но и для множества задач. После обучения генеративной модели на немаркированных данных – это называется предварительным обучением без учителя (unsupervised pre-training) –

модель просто настраивается на последующую задачу с помощью относительно небольшого количества данных, специфичных для задачи, что называется тонкой настройкой с учителем (supervised fine-tuning). Эта двухступенчатая схема широко используется в других моделях трансформеров, где за предварительным обучением без учителя следует точное дообучение на размеченных данных для конкретной задачи.

Чтобы архитектура GPT оставалась как можно более общей, в соответствии с конкретными задачами преобразуются только входные данные, в то время как вся архитектура остается почти неизменной. В соответствии с этим подходом выполняется преобразование (трансформирование) текстового ввода в упорядоченную последовательность в соответствии с задачей, чтобы предварительно обученная модель могла понять задачу из нее. Левая часть рис. 4.2 (подготовленного по мотивам оригинальной статьи) иллюстрирует архитектуру трансформера и цели обучения. В правой части показано, как трансформировать ввод для тонкой настройки на несколько задач.

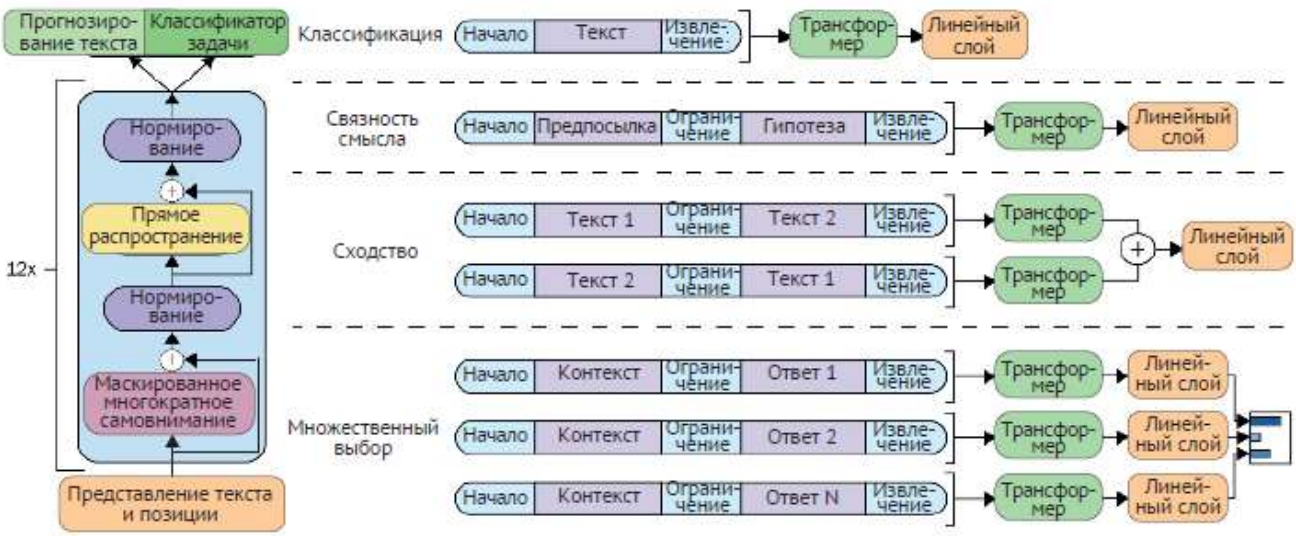

Рис. 4.2. Трансформация текстового ввода

GPT и две последующие модели этого типа в основном сосредоточились на поиске конкретной архитектуры, для которой не нужен этап тонкой настройки. Архитектура основана на идее, что модель может быть очень квалифицированной в том смысле, что она способна изучить большую часть информации о языке на этапе предварительной подготовки, поэтому на этапе тонкой настройки для нее остается мало работы. Таким образом, для большинства задач процесс тонкой настройки может быть завершен в течение трех эпох и с относительно небольшими выборками данных. В идеальном случае обучение без подготовки позволяет исключить этап тонкой настройки. Основная идея заключается в том, что модель может извлечь достаточно много информации о языке во время

предварительного обучения. Это особенно актуально для всех трансформерных моделей.

Последователи оригинальной GPT

GPT-2, преемник исходной GPT-1, представляет собой более крупную модель, обученную на гораздо большем количестве данных под названием WebText, чем ее предшественница. Она достигла самых передовых результатов в семи из восьми задач в условиях без подготовки, в которых не применялась тонкая настройка, но имела лишь ограниченный успех в некоторых других задачах. Она добилась сопоставимых результатов на небольших наборах данных в задаче выявления зависимости на большой протяженности. Авторы GPT-2 утверждали, что языковые модели не обязательно нуждаются в наборе явно размеченных данных для изучения задачи. Вместо этого они могут изучить такие задачи при обучении на огромном и разнообразном наборе данных веб-страниц. Это общий подход, согласно которому цель обучения $P(\text{выход} \mid \text{вход})$ в исходной GPT заменяют на $P(\text{выход} \mid \text{вход}, \text{задача } i)$, где модель производит разные выходные данные для одного и того же входа в зависимости от конкретной задачи – то есть

GPT-2 изучает несколько задач, обучая одну и ту же модель на неразмеченных данных. Одна предварительно обученная модель получает разные способности только через цель предварительного обучения. Мы встречаем аналогичные формулировки относительно параметров многозадачности и метазадач и в других исследованиях. Такой переход к многозадачному обучению (multi-task learning, MTL) позволяет решать множество разных задач для одного и того же ввода. Но откуда модели знают, какую задачу выполнять? Для этого и нужен перенос модели на задачу без подготовки (zero-shot task transfer)

По сравнению с исходной GPT, модель GPT-2 не имеет тонкой настройки для конкретной задачи и может работать без подготовки к новой задаче, когда все последующие задачи являются частью прогнозирования условных вероятностей (где условием является конкретная задача). Задача каким-то образом фигурирует во входных данных, и ожидается, что модель будет понимать природу последующих задач и давать соответствующие ответы. Например, для задачи машинного перевода с английского на турецкий язык обучение модели зависит не только от ввода, но и от задачи. Ввод организован таким образом, что за предложением на английском языке через разделитель следует предложение на турецком языке, по которому модель понимает, что задача является переводом с английского на турецкий.

Команда OpenAI обучила модель GPT-3 с 175 млрд параметров, что в 100 раз больше, чем у GPT-2. Архитектуры GPT-2 и GPT-3 похожи, но основные различия заключаются в размере модели и количестве/качестве набора данных. Из-за огромного количества данных в наборе и большого количества параметров, по которым она обучается, GPT-3 достигла лучших результатов по многим последующим задачам в условиях нулевой, однократной и многократной ($K = 32$) подготовки без какой-либо тонкой настройки на основе градиента. Команда разработчиков GPT-3 показала, что производительность модели увеличивается по мере увеличения размера параметра и количества примеров для многих задач, включая перевод, ответы на вопросы (QA) и задачи с маскированными токенами.

Transformer-XL

Модели на основе трансформеров страдают от контекста фиксированной длины из-за отсутствия рекуррентности в архитектуре и фрагментации контекста, хотя они способны к изучению протяженных зависимостей. Большинство трансформеров разбивают документы на список сегментов фиксированной длины, в которых перетекание информации между сегментами невозможно.

Следовательно, языковые модели не могут отслеживать протяженные зависимости, превышающие этот предел фиксированной длины. Более того, процедура сегментации строит сегменты, не обращая внимания на границы предложения. Сегмент может абсурдно состоять из второй половины предыдущего предложения и первой половины последующего, поэтому языковые модели могут пропустить необходимую контекстную информацию при прогнозировании следующего токена. В исследованиях эта проблема упоминается как проблема фрагментации контекста (context fragmentation problem).

Для решения этой проблемы авторы модели Transformer-XL предложили новую архитектуру трансформера, включая рекуррентный механизм на уровне сегментов и новую схему позиционного кодирования. Такой подход лег в основу многих последующих моделей. Он не ограничивается двумя последовательными сегментами, поскольку эффективный контекст может выходить за пределы двух сегментов. Рекуррентный механизм работает между каждыми двумя последовательными сегментами, что позволяет в определенной степени охватить несколько сегментов. Максимально возможная длина зависимости, которую может поддерживать модель, ограничена количеством слоев и длиной сегментов.

XLNet

Языковое моделирование с маскированием доминировало на этапе предварительного обучения моделей на основе трансформеров. Однако оно подверглось критике, поскольку замаскированные токены присутствуют на этапе

предварительного обучения, но отсутствуют на этапе тонкой настройки, что приводит к несоответствию между предварительным обучением и тонкой настройкой. Из-за этого модель не сможет использовать всю информацию, полученную на этапе предварительного обучения. В модели XLNet подход MLM (моделирование маскированного языка) заменили на языковое моделирование с пермутацией (permuted language modeling, PLM), которое представляет собой случайную перестановку (пермутацию) входных токенов для преодоления этого узкого места. Языковое моделирование с пермутацией заставляет каждую позицию токена использовать контекстную информацию со всех позиций, что приводит к изучению двунаправленного контекста.

Целевая функция только переставляет порядок разложения и определяет порядок предсказаний токенов, но не меняет естественные позиции последовательностей. Вкратце: модель выбирает некоторые токены в качестве цели после перестановки, а затем пытается предсказать их в зависимости от оставшихся токенов и естественного положения целевого предложения. Это позволяет использовать модель AR в двунаправленном режиме.

XLNet использует модели AE и AR. На самом деле это обобщенная модель AR; однако она может рассматривать токены как из левого, так и из правого контекстов благодаря языковому моделированию с пермутацией. Помимо своей целевой функции, XLNet включает в себя еще два важных механизма: рекуррентный механизм на уровне сегментов Transformer-XL и тщательно разработанный механизм двухпотокового внимания для ориентированных на цель представлений.

Модели Seq2Seq

Левая часть трансформера (энкодер) и его правая часть (декодер) связаны механизмом перекрестного внимания, что помогает каждому слою декодера следить за последним уровнем энкодера. Это естественным образом подталкивает модели к генерации выходных данных, которые тесно связаны с исходными входными данными. Модель Seq2Seq, которая относится к исходным трансформерам, достигает этого с помощью следующей схемы:

Входные токены→представления→энкодер→декодер→выходные токены

В моделях категории Seq2Seq энкодер и декодер остаются частью трансформера. Среди популярных моделей Seq2Seq можно упомянуть T5, двунаправленный авторегрессивный трансформер (bidirectional and auto-regressive transformer, BART) и предварительное обучение на предложениях с извлеченными пробелами для моделей абстрактного суммирования «последовательность–последовательность» (pre-training with extracted gap-

sentences for abstractive summarization sequence-to-sequence models (предварительное обучение с извлечением пробельных предложений для моделей абстрактного суммирования от последовательности к последовательности), PEGASUS).

T5

Большинство архитектур NLP, от Word2Vec до трансформеров, изучают представления и другие параметры, предсказывая замаскированные слова с использованием контекстных (соседних) слов. Мы относимся к задачам NLP как к задачам предсказания слов. В некоторых исследованиях почти все задачи NLP рассматриваются как QA (ответ на вопрос), или классификация токенов. Аналогичным образом модель T5 предложила общую структуру для решения многих задач, трансформируя их в задачу преобразования текста в текст. Идея, лежащая в основе T5, состоит в том, чтобы трансформировать все задачи NLP в проблему преобразования текста в текст (Seq2Seq), где и вход, и выход представляют собой список токенов, потому что структура преобразования текста в текст хорошо зарекомендовала себя при применении одной и той же модели к разнообразным задачам NLP от QA до резюмирования текста.

На рис. 4.3, показано, как T5 решает четыре разные задачи NLP – машинный перевод, лингвистическую приемлемость, семантическое сходство и резюмирование – в рамках единой структуры.

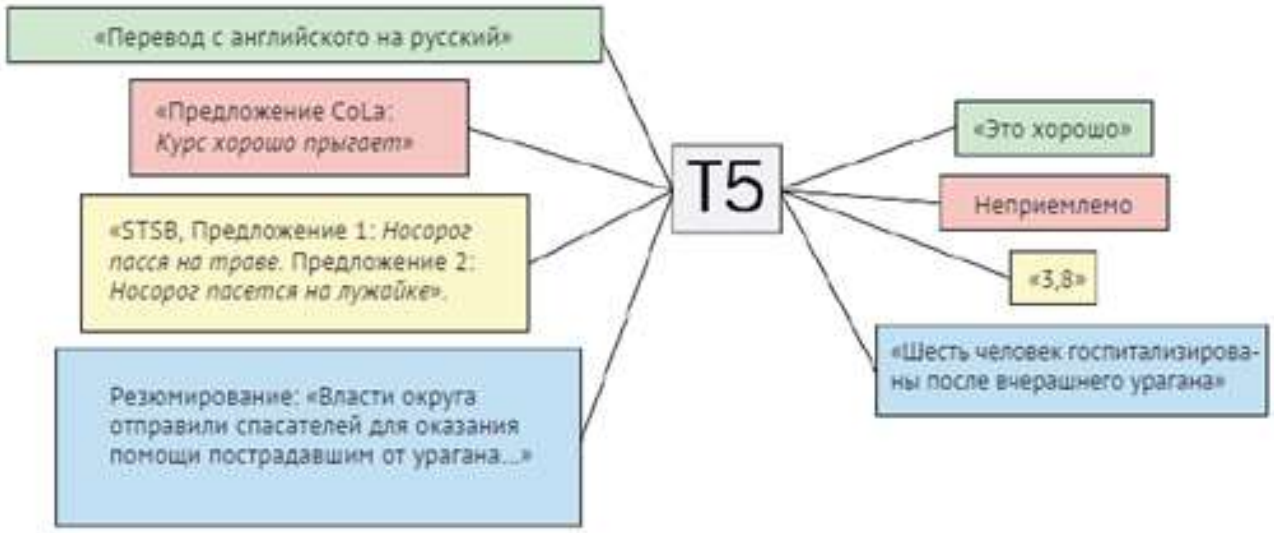

Рис. 4.3. Обобщенная схема функционирования T5

Модель T5 примерно соответствует исходной модели трансформера с энкодером-декодером. Изменения коснулись схемы слоев нормализации и представления позиций. Вместо использования синусоидального представления позиций или выученного представления T5 использует относительное позиционное представление, которое становится все более распространенным в

архитектурах трансформеров. T5 – это единая модель, которая может работать с разнообразным набором задач, например с генерацией языков. Что еще более важно, она приводит все исходные задачи к задаче преобразования текста в текст. В модель подают текстовый ввод, состоящий из префикса задачи и входных данных. Мы конвертируем помеченный текстовый набор данных в формат {'inputs': '...', 'targets': ...'}, где вставляем цель во входные данные в качестве префикса. Затем мы обучаем модель на размеченных данных, чтобы она узнала, что и как делать. Как показано на рис. 4.3, для задачи перевода с английского на русский используется параметр «Перевести с английского на русский: This is good». Такой ввод сгенерирует вывод «Это хорошо». Аналогичным образом любой ввод с префиксом «Резюмирование:» будет подвергаться резюмированию (сжатому изложению главного смысла).

BART

Как и в случае с XLNet, модель BART использует преимущества схем моделей AE и AR – стандартную архитектуру трансформера Seq2Seq с небольшой модификацией. BART – это предварительно обученная модель, для обучения которой применяли множество способов зашумления, искажающих исходный текст. Главный вклад модели BART в область NLP заключается в том, что она позволяет нам применять несколько схем искажения, как показано на рис. 4.4.

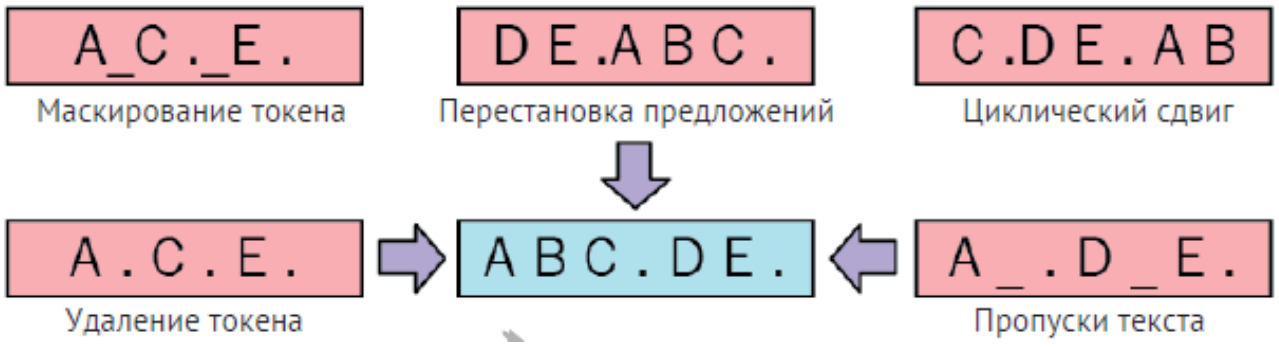

Рис. 4.4. Принцип работы модели BART

Рассмотрим каждую схему искажения, а именно:

- маскирование токенов: токены случайным образом маскируются символом [MASK], как и в модели BERT;
- удаление токенов: токены удаляются из документов случайным образом. Модель вынуждена определять, какие позиции удалены;
- пропуски текста: после SpanBERT производится выборка нескольких отрезков текста, а затем они заменяются одним токеном [MASK]. Также предусмотрена вставка токена [MASK];
- перестановка предложений: предложения во входных данных сегментируются и перемешиваются в случайном порядке;

- циклический сдвиг документа: документ сдвигается по кругу таким образом, что он начинается со случайно выбранного токена. На рис. 4.4 это токен C. Цель состоит в том, чтобы найти начальную позицию документа.

Для модели BART можно выполнить тонкую настройку несколькими способами для последующих приложений, таких как BERT. Для задачи классификации последовательностей входные данные проходят через энкодер и декодер, и окончательное скрытое состояние декодера считается изученным представлением. Затем на его основе простой линейный классификатор может делать прогнозы. Аналогичным образом для задач классификации токенов весь документ подается в энкодер и декодер, и окончательное состояние последнего декодера является представлением для каждого токена. Основываясь на этих представлениях, мы можем решить задачу классификации токенов. Задачи распознавания именованных объектов (NER) и частей речи (POS) могут быть решены с использованием этого окончательного представления, где NER идентифицирует в тексте объекты, такие как лицо и организация, а POS связывает каждый токен с лексической категорией, такой как существительное, прилагательное и т. д.

Если стоит задача генерации последовательности, то блок декодера модели BART, который является декодером AR, может быть напрямую настроен для задач генерации последовательности, таких как QA или резюмирование. Авторы BART (Льюис, Майк и др.) обучили модели, используя два стандартных набора данных резюмирования: CNN/DailyMail и XSum. Авторы также показали, что можно использовать как часть энкодера, которая применяет исходный язык, так и часть декодера, которая производит слова на целевом языке в качестве единого предварительно обученного декодера для машинного перевода. Они заменили уровень представления энкодера новым энкодером с произвольной инициализацией, чтобы изучать слова на исходном языке. Затем модель обучается сквозным способом, который обучает новый энкодер отображать иностранные слова во входные данные, которые BART может очищать от шума уже на целевом языке. Новый энкодер может использовать отдельный словарь, включая иностранный язык, из исходной модели BART.

Классификация текста

Классификация текста (также известная как категоризация текста) – это способ сопоставления документа (предложения, публикации, главы книги, содержимого электронной почты и т. д.) с категорией из предопределенного списка (классов). В случае двух классов, которые имеют положительные и отрицательные метки, это называется бинарной классификацией (binary

classification), или, если использовать более строгие термины, анализом тональности (sentiment analysis) или анализом эмоциональной окраски текста. Если классов больше двух, это называется многоклассовой классификацией (multi-class classification), когда классы являются взаимоисключающими, или классификацией с несколькими метками (multi-label classification), когда классы не являются взаимоисключающими и документ может получать более одной метки. Например, контент новостной статьи может быть связан со спортом и политикой одновременно. Помимо этой классификации, мы можем захотеть оценить документы в диапазоне [–1,1] или ранжировать их в диапазоне [1–5]. Мы можем решить эту проблему с помощью регрессионной модели, в которой вывод является числовым, а не категориальным.

К счастью, трансформеры позволяют нам эффективно решать эти проблемы. Для задач, связанных с парой предложений, таких как схожесть документов или выявление следствия, вводом является не одно предложение, а два предложения, как показано на рис. 4.5.

Рис. 4.5. Схема классификации текста

Мы можем оценить, насколько два предложения семантически похожи, или предсказать, являются ли они семантически похожими. Еще одна задача, связанная с парами предложений, – это выявление следствия, когда смысл второго предложения логически вытекает из первого. Такая задача относится к многоклассовой классификации. Например, в тесте GLUE на рис. 4.5. используются две последовательности, оцениваемые по классам следует/противоречит/нейтрально.

Представление предложений

Предварительно обученные модели BERT не обеспечивают эффективное и независимое представление предложений, поскольку всегда нуждаются в тонкой настройке. Это связано с тем, что мы можем рассматривать предварительно обученную модель BERT как неделимое целое, а семантика распространяется на все уровни, а не только на последний уровень. Без тонкой настройки независимое

использование ее внутренних представлений может оказаться неэффективным. Подобным моделям плохо даются задачи, связанные с обучением без учителя, такие как кластеризация, тематическое моделирование, поиск информации или семантический поиск. Например, модели приходится оценивать множество пар предложений во время задач кластеризации, а это вызывает огромные вычислительные затраты.

К счастью, на основе исходной модели BERT было создано множество модификаций, таких как Sentence (предложение) BERT8 (SBERT), для получения семантически значимых и независимых представлений предложений. В литературе по NLP было предложено множество нейросетевых методов представления предложений для отображения одного предложения в общее пространство признаков (модель векторного пространства), в котором функция косинуса (или скалярное произведение) обычно используется для измерения сходства, а евклидово расстояние – для измерения различия. Ниже перечислены некоторые приложения, которые можно эффективно решить с помощью представления предложений:

- задачи поиска пары предложений;
- поиск информации;
- ответы на вопросы;
- обнаружение повторяющихся вопросов;
- обнаружение перефразирования;
- кластеризация документов;
- тематическое моделирование.

Самый простой, но наиболее эффективный вид представления предложений – это операция объединения средних значений (average-pooling), которая выполняется при представлении слов, входящих в предложение. Некоторые ранние нейросетевые методы – например, Doc2Vec, Skip-Thought, FastSent и Sent2Vec – изучали представление предложений с обучением без учителя. Подход Doc2Vec использовал теорию распределения на уровне токенов и целевую функцию для предсказания соседних слов, аналогично Word2Vec. Этот подход вводит дополнительный токен памяти (под названием Paragraph-ID) в каждое предложение, что напоминает токены CLS или SEP в библиотеке transformers. Этот дополнительный токен действует как часть памяти, которая хранит представления контекста или документа. SkipThought и FastSent считаются подходами на уровне предложения, где целевая функция используется для предсказания смежных предложений. Эти модели извлекают значение

предложения, чтобы получить необходимую информацию из соседних предложений и их контекста.

Некоторые другие методы, такие как InferSent, основаны на обучении с учителем и многозадачном переносе обучения для изучения общих представлений предложений. InferSent обучил различные контролируемые задачи для более эффективного внедрения. Обучаемые с учителем модели на основе RNN, такие как GRU или LSTM, используют для получения представления предложений последнее скрытое состояние (или уложенные в стек полные скрытые состояния).

Кросс-энкодер и биэнкодер

До сих пор мы обсуждали, как обучать языковые модели на основе трансформеров и выполнять их тонкую настройку при частичном и полном обучении с учителем соответственно, и добились успешных результатов благодаря трансформерной архитектуре. После того как тонкий линейный слой для конкретной задачи помещен поверх предварительно обученной модели, все веса сети (а не только последний тонкий слой для конкретной задачи) настраиваются с помощью данных, размеченных для конкретной задачи. Как настроить модель BERT для двух разных групп задач (одно предложение или пара предложений) без каких-либо изменений в архитектуре. Единственное отличие состоит в том, что для задач, связанных с парами предложений, входные предложения объединяют и помечают токеном [SEP]. Таким образом, самовнимание применяется ко всем токенам объединенных предложений. Это большое преимущество модели BERT, где оба входных предложения могут получать необходимую информацию друг от друга на каждом уровне. В конце концов, они кодируются одновременно. Этот процесс называется перекрестным, или гибридным, кодированием, а модель – кросс-энкодером (cross-encoder).

Однако у кросс-энкодеров есть два недостатка, которые были отмечены авторами SBERT, а именно:

- кросс-энкодер неудобен для многих задач, связанных с парами предложений, из-за того, что необходимо обработать слишком много возможных комбинаций. Например, чтобы получить два самых близких предложения из 1000 предложений, модель кросс-энкодера (BERT) должна выполнить около $500\,000$ $(n \times (n-1)/2)$ вычислений вывода. Следовательно, она будет очень медленной по сравнению с альтернативными решениями, такими как SBERT или USE (Universal Sentence Encoder – универсальный кодировщик предложений). Эти альтернативные модели генерируют независимые представления предложений, к которым можно легко применить метрику подобия (косинусное сходство) или метрику несходства (евклидову или манхэттенскую). Заметим, что вычисление

этих метрик может быть эффективно реализовано на современных архитектурах. Более того, с помощью оптимизированной индексной структуры мы можем снизить вычислительную сложность с многих часов до нескольких минут при сравнении или кластеризации множества документов;

- из-за необходимости в обучении с учителем модель BERT не может выводить независимые значимые представления предложений. Предварительно обученную модель BERT трудно использовать для таких задач, как кластеризация, семантический поиск или тематическое моделирование. Модель BERT создает вектор фиксированного размера для каждого токена в документе. При обучении без учителя представление на уровне документа может быть получено путем усреднения или объединения векторов токенов, а также токенов SEP и CLS. Позже мы увидим, что BERT генерирует векторное представление предложений с качеством ниже среднего и что его показатели производительности обычно хуже, чем у методов объединения представлений слов, таких как Word2Vec, FastText или GloVe.

Альтернативой кросс-энкодеров являются биэнкодеры (такие как SBERT), независимо отображающие пару предложений в семантическое векторное пространство, как показано на рис. 4.6. Поскольку представления являются раздельными, биэнкодеры могут кешировать закодированное представление для каждого ввода, что сокращает время вывода. SBERT – одна из успешных модификаций системы BERT с биэнкодером. SBERT модифицирует модель BERT для получения семантически значимых и независимых вложений предложений.

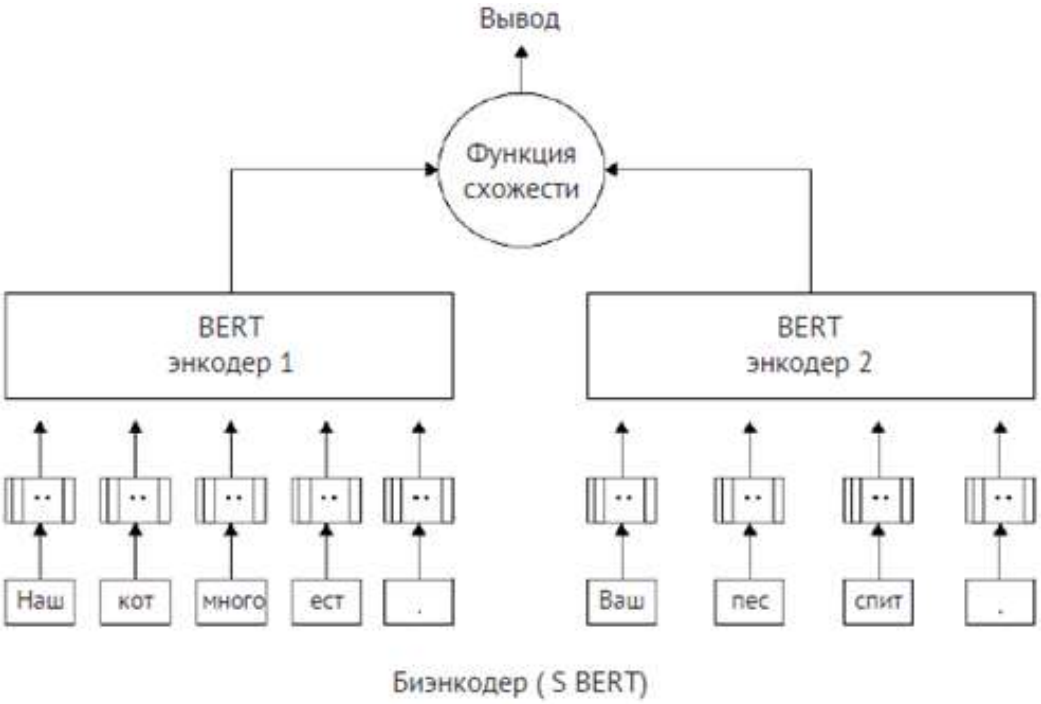

Рис. 4.6. Архитектура биэнкодера

Разреженное внимание с фиксированными паттернами

Напомним, что механизм внимания состоит из запроса, ключа и значений, как в этой условной формуле:

$$Attention(Q, K, V) = Score\,(Q, K) \bullet V$$

Здесь функция Score (Счет чаще всего softmax) выполняет QK^{T}-умножение, которое требует $O(n^2)$ памяти и вычислительной сложности, поскольку в режиме полного самовнимания позиция токена соотносится с позициями всех остальных токенов для построения представлений своей позиции. Мы повторяем тот же процесс для всех позиций токенов, чтобы получить их представления, что приводит к проблеме квадратичного роста сложности. Это очень дорогой способ обучения, особенно для задач NLP с протяженным контекстом. Естественно задать вопрос: нужно ли нам такое плотное взаимодействие или есть более дешевый способ проведения вычислений? Многие исследователи обращались к этой проблеме и использовали различные методы, чтобы избавиться от квадратично возрастающей сложности механизма самовнимания. В основном они искали компромисс между производительностью, вычислениями и памятью, особенно для длинных документов.

Самый простой способ уменьшить сложность – это разбить полную матрицу самовнимания или найти другой более дешевый способ аппроксимировать полное внимание. Паттерны разреженного внимания показывают, как подключать/отключать определенные позиции, не нарушая потока информации через слои, что помогает модели отслеживать долгосрочную зависимость и выполнять кодирование на уровне предложений.

Полное самовнимание и разреженное внимание схематически показаны на рис. 4.7, где строки соответствуют выходным позициям, а столбцы – входным.

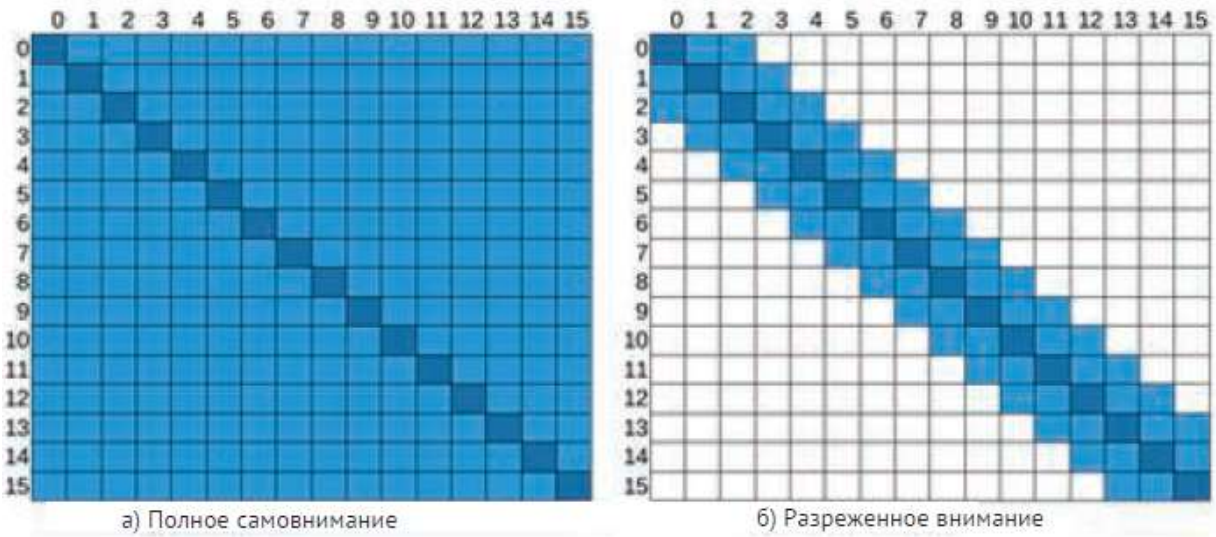

Рис. 4.7. Полное самовнимание и разреженное внимание

Модель полного самовнимания будет напрямую передавать информацию между любыми двумя позициями. С другой стороны, в механизме внимания с локальным скользящим окном, которое является разреженным вниманием, как показано справа на рисунке, пустые ячейки означают отсутствие взаимодействия между соответствующими позициями ввода-вывода. Разреженная модель на рисунке основана на фиксированных паттернах, которые представляют собой определенные правила, разработанные вручную. Предполагается, что полезная информация находится в каждой соседней позиции относительно текущей. Каждый токен запроса относится к ключевым токенам window/2 слева и ключевым токенам window/2 справа от этой позиции. В следующем примере размер окна выбран равным 4. Это правило одинаково применяется ко всем слоям трансформера. В некоторых исследованиях размер окна увеличивается по мере продвижения по слоям.

В разреженном режиме информация передается через подключенные узлы (непустые ячейки) модели. Например, выходная позиция 7 матрицы разреженного внимания не может напрямую соответствовать входной позиции 3 (см. разреженную матрицу справа на рис. 4.7), поскольку ячейка (7,3) считается пустой. Однако позиция 7 косвенно связана с позицией 3 через позицию токена 5, то есть (7–>5, 5–>3 => 7–>3). На рисунке видно, что в то время как полное самовнимание требует n^2 активных ячеек (вершин), разреженной модели нужно примерно $5 \times n$.

Другой важный тип – это глобальное внимание (global attention). Несколько выбранных токенов или несколько принудительно внедренных токенов используются в качестве глобального внимания, которое может уделять внимание всем другим позициям и быть объектом их внимания. Следовательно, максимальное расстояние пути между любыми двумя позициями токена равно 2. Предположим, у нас есть предложение [GLB, the, cat, is, very, sad], где Global (GLB) – это внедренный глобальный токен, а размер окна равен 2, что означает, что объектом внимания токена может быть только непосредственный сосед слева или справа, а также GLB. Между токенами cat и sad нет прямого взаимодействия. Но мы можем отследить взаимодействия cat–>GLB, GLB–>sad, которые образуют гиперссылку через токен GLB. Глобальные токены можно выбрать из существующих токенов или добавить как (CLS). Как показано на рис. 4.8, первые две позиции токенов выбраны как глобальные токены:

Кстати, эти глобальные токены не обязательно должны быть в начале предложения. Например, модель Longformer случайным образом выбирает глобальные токены в дополнение к первым двум токенам.

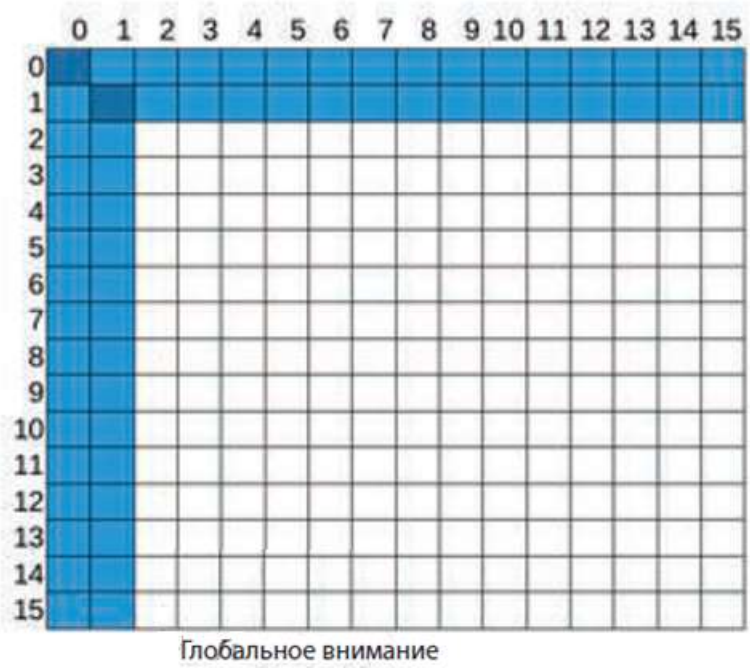

Рис. 4.8. Глобальное внимание

Есть еще четыре широко распространенных паттерна. Случайное внимание (random attention, первая матрица на рис. 4.9) используется для облегчения потока информации путем случайного выбора из существующих токенов.

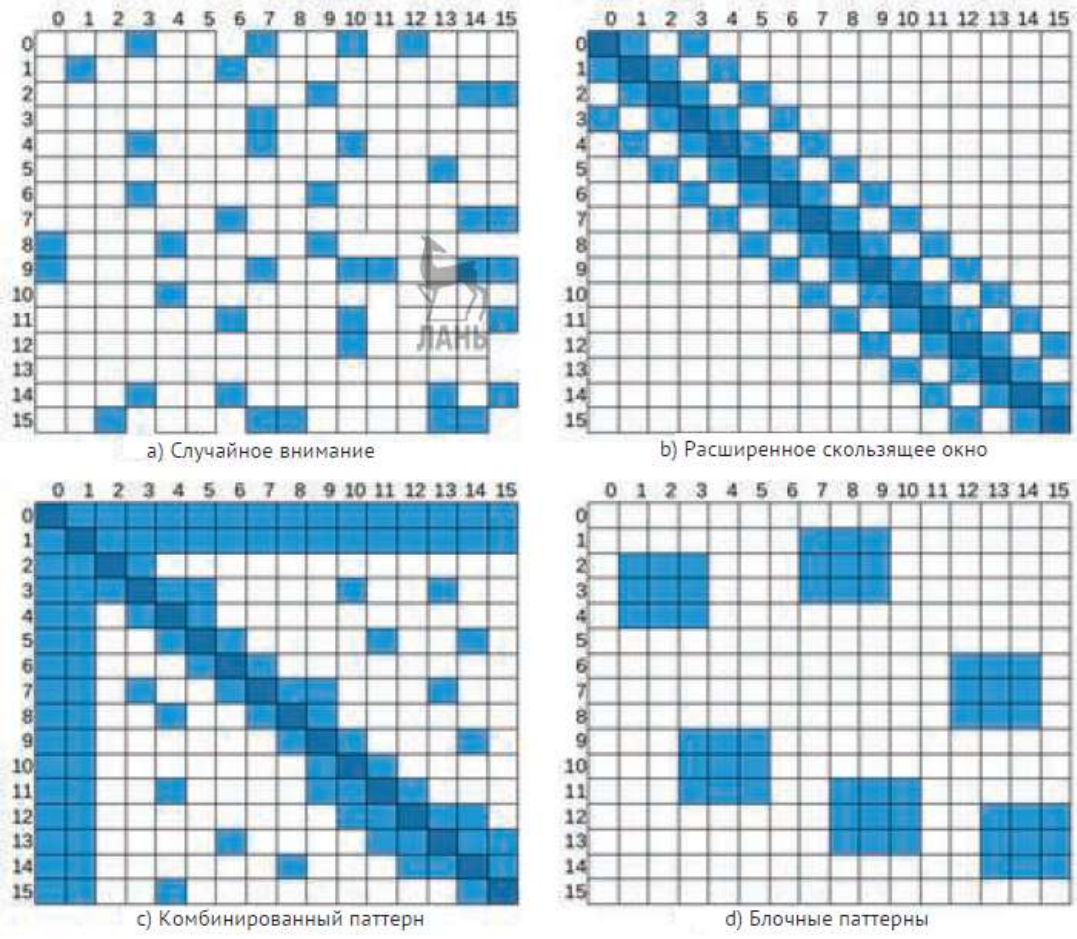

Рис. 4.9. Случайное, расширенное, комбинированное и блочное внимание

Но в большинстве случаев мы используем случайное внимание как часть комбинированного паттерна (combined pattern, нижняя левая матрица), который состоит из комбинации других моделей. Расширенное внимание (dilated attention) похоже на скользящее окно, но в окне есть пробелы, как показано в правом верхнем углу рис. 8.6.

Блочный паттерн (blockwise pattern, нижний правый угол) обеспечивает основу для других паттернов. Он разбивает токены на фиксированное количество блоков, что особенно полезно для задач с длинным контекстом. Например, когда матрица внимания 4 096×4 096 разбивается на блоки размером 512, то формируются 8 блоков запроса и ключевых блоков размером 512×512. Многие эффективные модели, такие как BigBird и Reformer, в основном разбивают токены на блоки, чтобы уменьшить сложность.

Еще есть обучаемые паттерны (learnable pattern) являются альтернативой фиксированным (предопределенным) паттернам. Эти методы извлекают паттерны, исключительно исходя из данных, без обучения с учителем. Они используют некоторые приемы для измерения сходства между запросами и ключами для их правильной кластеризации. Это семейство трансформеров сначала изучает, как кластеризовать токены, а затем ограничивает взаимодействие, чтобы получить оптимальное представление матрицы внимания.

Разложение низкого ранга, методы ядра и другие подходы

Последняя тенденция в области разработок эффективных моделей – использование низкоранговых аппроксимаций полной матрицы самовнимания. Эти модели считаются самыми легкими, поскольку могут уменьшить сложность самовнимания с $O(n^2)$ до $O(n)$ как по времени вычислений, так и по объему памяти. При выборе очень маленькой размерности проекции k, такой, что $k << n$, значительно уменьшается потребление памяти и дискового пространства. Linformer и Synthesizer – это модели, которые эффективно аппроксимируют полное внимание с помощью разложения низкого ранга. Они разлагают скалярное произведение $N \times N$ механизма внимания исходного преобразователя через линейные проекции.

Ядро внимания (kernel attention) – это еще одно новое семейство методов для повышения эффективности, рассматривающее механизм внимания с точки зрения ядра. Ядро – это функция, которая принимает два вектора в качестве аргументов и возвращает результат их проекции с картой распределения признаков (feature map). Она позволяет нам работать в многомерном пространстве признаков, даже не вычисляя координаты данных в этом многомерном пространстве, потому что вычисления в этом пространстве становятся более

дорогостоящими. Здесь в игру вступает трюк с ядром. Эффективные модели, основанные на ядре, позволяют нам переписать механизм самовнимания, чтобы избежать явного вычисления матрицы $N \times N$. В машинном обучении наиболее известный алгоритм на основе функции ядра – это машина опорных векторов, где широко применяют ядро радиальной базисной функции или полиномиальное ядро, особенно для нелинейности. Среди трансформеров наиболее яркими представителями этой категории являются архитектуры Performer и Linear Transformer.