ЛЕКЦИЯ 3. Языковые модели на основе автоэнкодеров

BERT – одна из языковых моделей на основе автоэнкодера

Нейросетевая модель-трансформер BERT (Bidirectional Encoder Representations from Transformers – Представления двунаправленного кодера от трансформаторов) была одной из первых языковых моделей автоматического кодирования, в которых использовался стек энкодера Transformer с небольшими изменениями для языкового моделирования.

Архитектура BERT – это многослойный энкодер, основанный на оригинальной реализации модели Transformer. Сама модель Transformer изначально предназначалась для задач машинного перевода, но основным улучшением, представленным в BERT, является использование этой части архитектуры для обеспечения улучшенного языкового моделирования. Эта языковая модель после предварительного обучения способна обеспечить глобальное понимание языка, на котором она обучается.

Задачи предварительного обучения языковой модели BERT

Чтобы иметь четкое представление о применяемом в BERT маскированном языковом моделировании, давайте рассмотрим его более подробно. Маскированное языковое моделирование (masked language modeling) – это задача обучения модели на входных данных (предложения, в которых одно из слов замаскировано токеном [MASK]) и получение вывода в виде предложений, в которых вместо маски подставлены правильные слова. Но каким образом это помогает модели получать лучшие результаты в последующих задачах, таких как классификация? Ответ прост: если модель может выполнять закрытый тест (лингвистический тест для оценки понимания языка путем заполнения пробелов), то она имеет общее понимание языка в целом. Поэтому она будет работать лучше после обучения конкретным задачам по сравнению с другими моделями.

Вот пример закрытого теста:

Джордж Вашингтон был первым президентом _____ Штатов Америки.

Ожидается, что вместо пробела будет подставлено слово Соединенных. Перед маскированной языковой моделью стоит та же задача, и требуется подставить вместо маски правильные токены. Однако позиция замаскированного токена в предложении выбирается случайным образом.

Еще одна задача, на которой обучается BERT, – это прогнозирование следующего предложения. Эта задача предварительного обучения гарантирует, что BERT изучает не только отношения между всеми токенами при прогнозировании замаскированных токенов, но также старается понять отношения между двумя предложениями. При обучении мы выбираем пару

предложений и передаем в BERT с токеном-разделителем [SEP] между ними. Из набора данных так-же известно, идет ли второе предложение после первого или нет.

Ниже приводится пример задачи NSP:

Испытуемый должен заполнить пропуски. Цена биткойнов слишком высока по сравнению с другими альткойнами.

В этом примере требуется, чтобы модель предсказала отрицательный результат (предложения не связаны друг с другом).

Эти две задачи предварительного обучения позволяют BERT понимать сам язык. Представление токенов BERT обеспечивает контекстное представление для каждого токена. Контекстное представление (contextual embedding) означает, что каждый токен имеет представление, полностью связанное с окружающими токенами. В отличие от Word2Vec и подобных моделей, BERT обладает лучшей информацией о каждом представлении токена. С другой стороны, задачи NSP позволяют BERT иметь лучшие представления токенов [CLS (токен классификации)]. Этот токен, как было сказано в первой главе, предоставляет информацию обо всем вводе. [CLS] используется для задач классификации и на этапе предварительного обучения изучает общее представление всего ввода. На рис. 3.1 показаны соответствующие входные и выходные данные обобщенного представления модели BERT:

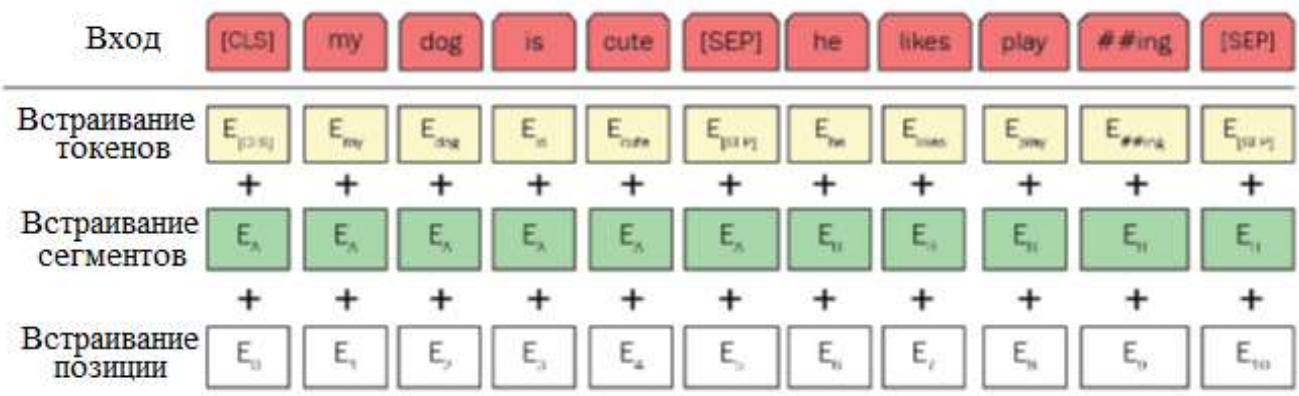

Рис. 3.1. Обобщенное представление модели BERT

Устройство модели BERT

Токенизаторы – одна из самых важных частей многих приложений NLP. Для BERT применяется токенизация WordPiece. Стоит отметить, что WordPiece, SentencePiece и BytePairEncoding (BPE) являются тремя наиболее широко известными токенизаторами, используемыми в различных архитектурах на основе трансформеров. Мы еще вернемся к ним в следующих разделах. Основная причина того, что BERT или любая другая трансформерная архитектура

использует токенизацию частей слов, – это способность таких токенизаторов работать с неизвестными токенами.

BERT также использует позиционное кодирование, чтобы гарантировать, что модель учитывает положение токенов. BERT и подобные модели не учитывают порядок входной последовательности.

Обычные модели, такие как LSTM и RNN, получают позицию в соответствии с порядком токенов в последовательности. Чтобы донести эту дополнительную информацию в BERT, применяется позиционное кодирование.

Предварительное обучение BERT предоставляет модели обобщенную информацию о языке, но на практике при решении различных задач, таких как классификация последовательностей, классификация токенов или ответы на вопросы, используются разные части вывода модели.

Например, в случае задачи классификации последовательности, такой как анализ тональности или классификация предложений, для BERT предлагается использовать представление [CLS] из последнего уровня. Однако есть другие исследования, в которых проводят классификацию при помощи различных методов с использованием BERT (с использованием усредненного представления токенов на основе всех токенов, развертывания LSTM на последнем уровне или даже использования CNN поверх последнего уровня). Представление [CLS] на последнем уровне для классификации последовательностей может использоваться любым классификатором, но чаще всего он представляет собой плотный слой с размером ввода, равным размеру представления окончательного токена, и размером вывода, равным количеству классов с функцией активации Softmax. Альтернативным вариантом является использование сигмоидной функции, когда вывод может быть многозначным, а сама задача заключается в классификации с несколькими метками.

Чтобы более подробно рассказать вам о том, как на самом деле работает BERT, на рис. 3.2 представлен пример задачи NSP. Здесь для облегчения понимания применяется упрощенная токенизация.

Модель BERT встречается в различных вариантах с разными настройками. Например, можно менять размер входных данных. В предыдущем примере задано значение 512, и, следовательно, максимальный размер последовательности, которую модель может получить в качестве входных данных, равна 512. Однако в последовательность входят специальные токены [CLS] и [SEP], поэтому фактический размер данных уменьшается до 510. С другой стороны, использование WordPiece в качестве токенизатора дает на выходе токены частей слов, и после токенизации размер увеличится, потому что токенизатор разбивает

слова на части (как правило, если они не встречались в корпусе предварительного обучения).

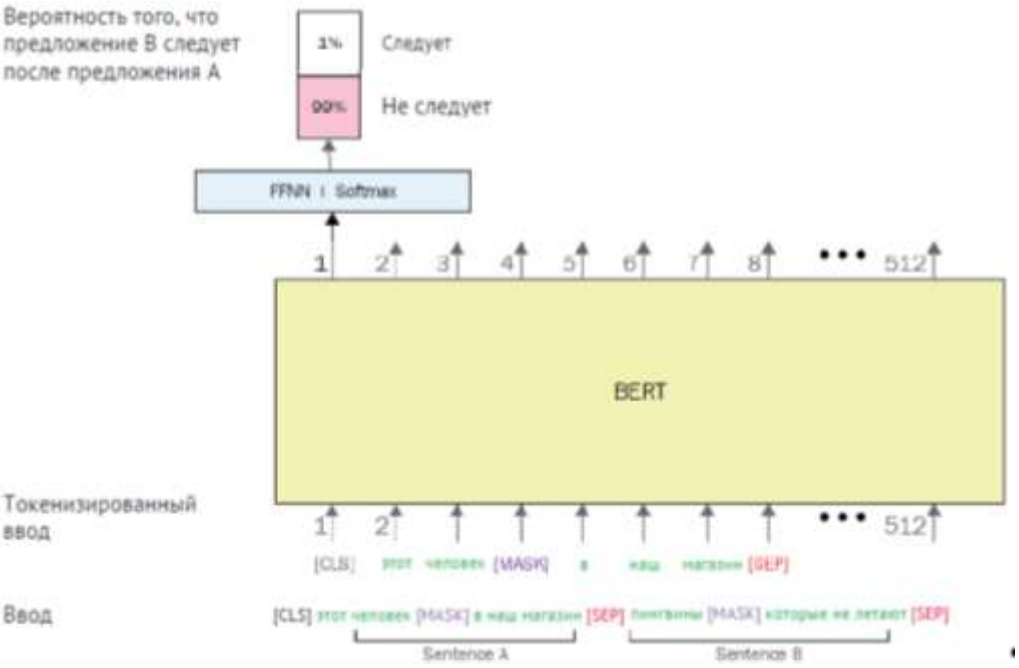

Рис. 3.2. Пример модели BERT для задачи NSP

На рис. 3.3 представлены схемы модели BERT для различных задач.

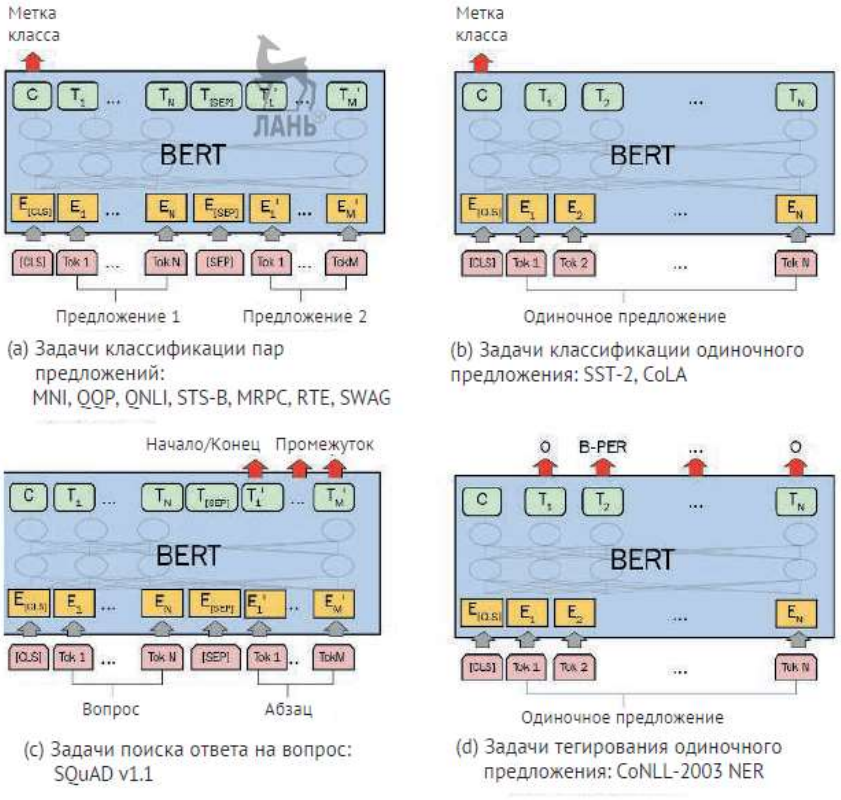

Рис. 3.3. Модель BERT для различных задач NLP

Для задачи NER вместо [CLS] используется вывод каждого токена. Если стоит задача ответа на вопрос, то вопрос и ответ объединяются при помощи токена-разделителя [SEP], а ответ аннотируется при помощи данных

Начало/Конец и Промежуток с последнего уровня. В данном случае Абзац – это контекст, в котором задается Вопрос.

Независимо от всех этих задач, наиболее важной способностью BERT является контекстное представление текста. Причина, по которой эта модель столь успешна в различных задачах, заключается в архитектуре энкодера Transformer, который представляет входные данные в виде плотных векторов. Эти векторы можно легко преобразовать в выходные данные с помощью очень простых классификаторов.

Обзор других моделей с автоэнкодером

Теперь рассмотрим альтернативные автоэнкодерные модели, которые в той или иной степени отличаются от исходной модели BERT. Эти альтернативные реализации привели к достижению лучших решений последующих задач за счет использования многих усовершенствований: оптимизации процесса предварительного обучения и количества слоев или проходов механизма внимания, повышения качества данных, разработки более совершенных целевых функций и т. д. Источники улучшений можно разделить на две категории: улучшенная архитектура и предварительная подготовка перед обучением.

В последнее время было предложено много эффективных альтернатив модели BERT, поэтому здесь невозможно охватить и объяснить их все. Мы можем лишь предложить обзор некоторых моделей из числа наиболее упоминаемых в литературе и наиболее часто используемых в тестах NLP. Начнем с модели ALBERT – еще одной реализации BERT, в которой особое внимание уделено выбору архитектуры.

ALBERT

Считается, что качество языковых моделей улучшается по мере увеличения их размера. Однако обучение таких моделей становится все более сложным из-за ограничений памяти и более длительного времени обучения. Чтобы решить эти проблемы, команда Google предложила модель ALBERT (A Lite BERT, упрощенная модель BERT), которая действительно является обновленной реализацией архитектуры BERT за счет использования нескольких новых методов, снижающих потребление памяти и повышающих скорость обучения. Обновление архитектуры привело к тому, что новые языковые модели масштабируются намного лучше, чем исходная модель BERT. Наряду с 18-кратным уменьшением количества параметров ALBERT обучается в 1,7 раза быстрее, чем исходная «большая» модель BERT.

Модель ALBERT выигрывает в основном за счет следующих трех модификаций первоисточника:

- факторизованная параметризация представления;
- межуровневое совместное использование параметров;
- оценка потерь связности между предложениями.

Первые две модификации – это методы уменьшения параметров, которые связаны с проблемой размера модели и потребления памяти в исходной модели BERT. Третья модификация означает новую целевую функцию – предсказание порядка предложений (sentence-order prediction (предсказание порядка предложений), SOP), заменяющее задачу предсказания следующего предложения (next sentence prediction, NSP) в BERT, что привело к гораздо более тонкой модели и повышению производительности.

Факторизованная параметризация представления используется для разложения большой матрицы представления словаря на две маленькие матрицы, которые отделяют размер скрытых слоев от размера словаря. Эта декомпозиция уменьшает параметры внедрения с $O(V \times H)$ до O(V×E+E×H), где V – словарь, H – размер скрытого слоя, E – представления, что приводит к более эффективному использованию общих параметров модели, если выполняется условие $H >> E$.

Межуровневое совместное использование параметров предотвращает увеличение общего числа параметров, по мере того как сеть становится глубже. Этот метод считается еще одним способом повышения эффективности параметров, поскольку мы можем поддерживать небольшой размер параметра путем совместного использования или копирования. Как декларируется авторы модели представили эксперименты со многими способами обмена параметрами, такими как совместное использование параметров только для прямого распространения между уровнями или совместное использование параметров только для механизма внимания и т. д.

Другая модификация модели ALBERT – оценка потери связности между предложениями. Как мы уже упоминали, архитектура BERT использует преимущества двух функций потерь: потери при маскированном языковом моделировании и NSP. При прогнозировании того, появятся ли два сегмента подряд в исходном тексте, NSP сопровождается двоичной кросс-энтропийной потерей. Отрицательные примеры получены путем выбора двух сегментов из разных документов. Однако команда разработчиков ALBERT раскритиковала NSP за то, что это фактически задача обнаружения темы, которая считается относительно простой задачей. Поэтому команда предложила функцию потерь, основанную в первую очередь на связности, а не на предсказании темы.

Они использовали потерю SOP (предсказание порядка предложений), которая сфокусирована на моделировании взаимосвязи между предложениями, а

не на предсказании темы. Потеря SOP использует те же положительные примеры, что и BERT (которые представляют собой два последовательных сегмента из одного и того же документа), а в качестве отрицательных примеров – те же два последовательных сегмента, но переставленные местами.

RoBERTa

Надежно оптимизированный подход к предварительному обучению BERT (Robustly Optimized BERT pre-training Approach (Надежно оптимизированный подход к предварительному обучению BERT), RoBERTa) – еще одна популярная реализация BERT. Благодаря ей удалось добиться более существенных улучшений в стратегии обучения, чем за счет доработки архитектуры. Она превзошла BERT почти по всем индивидуальным задачам в бенчмарке GLUE. Одним из первоначальных вариантов усовершенствования было динамическое маскирование. Хотя статическое маскирование лучше для некоторых задач, команда RoBERTa показала, что динамическое маскирование лучше отражается на общей производительности. Далее мы сравним обновленную модель с BERT и кратко обобщим изменения.

Изменения в архитектуре:

- отказ от цели обучения предсказанию следующего предложения;
- динамическое изменение шаблонов маскирования вместо статического маскирования, которое выполняется путем создания шаблонов маскирования всякий раз, когда они передают последовательность в модель;
- токенизатор частей слов BPE.

Изменения в обучении:

- развитие обучающих данных: используется больше данных, например 160 ГБ вместо 16 ГБ, изначально использовавшихся в BERT. В исследовании принимался во внимание не только размер данных, но и качество и разнообразие;
- более длинные итерации – до 500 000 шагов предварительного обучения;
- увеличенный размер пакета;
- более длинные последовательности, что требует меньшего паддинга (заполнения);
- больший словарный запас – 50 тыс. BPE вместо 30 тыс. BPE.

ELECTRA

Модель ELECTRA (предложенная Кевином Кларком и др. в 2020 году) фокусируется на новой маскированной языковой модели, использующей в качестве цели обучения обнаружение замены токенов. Во время предварительного обучения модель вынуждена научиться отличать настоящие

входные токены от синтетически сгенерированных замен, где синтетический отрицательный пример выбирается из вероятных, а не случайно выбранных токенов. Разработчики модели ALBERT критиковали цель NSP BERT за то, что это простая задача выявления темы, и за использование некачественных отрицательных примеров. Модель ELECTRA обучает две нейронные сети, генератор и дискриминатор, таким образом, что первая генерирует качественные отрицательные примеры, а вторая отличает исходный токен от замененного. Зная про сети GAN (Генеративно-состязательная нейросеть) из области компьютерного зрения, когда генератор G создает поддельные изображения и пытается обмануть дискриминатор D, а нейросеть дискриминатора пытается выявить обман. Модель ELECTRA применяет почти аналогичный подход генератора-дискриминатора для замены исходных токенов высококачественными отрицательными примерами, которые выглядят правдоподобно, но сгенерированы синтетически.

Использование алгоритмов токенизации

Итак мы рассмотрели обучение модель BERT с помощью специального токенизатора, а именно BertWordPieceTokenizer. Теперь стоит подробнее обсудить процесс токенизации. Токенизация – это способ разделения текстового ввода на токены и присвоения идентификатора каждому токену перед загрузкой в архитектуру нейронной сети. Самый интуитивно понятный способ – разделить последовательность на более мелкие части по пространственному критерию. Однако такие подходы не соответствуют требованиям некоторых языков, например японского, а также могут привести к огромным проблемам со словарем. Почти все варианты модели Transformer используют токенизацию частей слов не только для уменьшения размерности, но и для кодирования редких (или неизвестных) слов, не встречавшихся при обучении. Токенизация основана на идее, что каждое слово, включая редкие или неизвестные слова, можно разложить на значимые более мелкие фрагменты, которые являются широко распространенными символами в обучающем корпусе.

Некоторые традиционные токенизаторы, разработанные в Moses и представленные в библиотеке NLTK (набор инструментов для естественного языка), применяют передовые методы, основанные на правилах. Но алгоритмы токенизации, которые используются с Transformers, основаны на самоконтролируемом обучении и извлекают правила из корпуса. Простые интуитивно понятные решения для токенизации на основе правил основаны на использовании символов, знаков препинания или пробелов. Токенизация на основе символов приводит к тому, что языковые модели теряют смысловое значение ввода. Несмотря на то что это может уменьшить размер словаря, что

хорошо, модель затрудняет понимание значения слова cat с помощью кодирования символов c, a и t. Более того, размер входной последовательности становится очень большим. Точно так же модели на основе пунктуации не могут правильно обрабатывать некоторые выражения, такие как английские haven't или ain't.

В последнее время неотъемлемой частью архитектур Transformer стали несколько усовершенствованных алгоритмов токенизации частей слов, таких как BPE. Эти современные процедуры токенизации состоят из двух этапов: этап предварительной токенизации просто разбивает входные данные на токены, используя пробел или правила языка. Второй этап заключается в обучении токенизатора и создании базового словаря разумного размера на основе токенов.

Попарное кодирование байтов

Алгоритм попарного кодирования байтов (byte pair encoding, BPE) – это метод сжатия данных. Он сканирует последовательность данных и итеративно заменяет наиболее частую пару байтов одним символом. Впервые он был адаптирован и предложен в 2015 г., чтобы решить проблему неизвестных и редких слов в области машинного перевода. В настоящее время он успешно используется в GPT-2 и многих других современных моделях. Многие современные алгоритмы токенизации основаны на подобных методах сжатия.

Он представляет текст как последовательность символьных n-грамм, которые также называются подсловами (subwords) символьного уровня. Изначально обучение начинается со словаря всех символов Unicode, встречающихся в корпусе. Это маленький словарь для английского языка, но его объем резко возрастает для языков с большим количеством символов, таких как японский. Затем алгоритм итеративно вычисляет биграммы символов и заменяет наиболее частые из них специальными новыми символами. Например, в английских текстах нередко встречаются символы t и h. Заменим последовательность этих символов на символ th. Этот процесс продолжается итеративно до тех пор, пока словарный запас не достигнет желаемого объема. Самый распространенный размер словарного запаса составляет около 30 000 символов.

Алгоритм BPE особенно эффективен для представления неизвестных слов. Однако он не может гарантировать обработку редких слов и/или слов, включая редкие подслова. В таких случаях он заменяет редкие символы на специальный символ <UNK>, что может привести к утрате смысла слов. В качестве потенциального решения был предложен байтовый BPE (BBPE), который

использует 256-байтовый набор словаря вместо символов Unicode, чтобы гарантировать, что каждый базовый символ включен в словарь.

Токенизатор WordPiece

WordPiece – еще один популярный алгоритм сегментации слов, широко используемый в BERT, DistilBERT и Electra. Он был предложен Шустером и Накадзимой для решения проблемы голоса в японском и корейском языках в 2012 году. Причина, по которой этому вопросу было посвящено отдельное исследование, заключается в том, что сегментация слов важна для предварительной обработки многих азиатских языков, потому что в этих языках пробелы используются редко. Поэтому при изучении NLP на азиатских языках мы чаще сталкиваемся с подходами к сегментации слов. Подобно BPE, WordPiece использует большой корпус для изучения словарного запаса и правил объединения. В то время как BPE и BBPE изучают правила слияния на основе статистики совместной встречаемости, алгоритм WordPiece использует оценку максимального правдоподобия для извлечения правил слияния из корпуса. Сначала он инициализирует словарь с помощью символов Unicode, которые также называются словарными символами. Он обрабатывает каждое слово в обучающем корпусе как список символов (изначально символы Юникода), а затем итеративно генерирует новые символы путем объединения двух символов из всех возможных пар символов-кандидатов на основе максимизации правдоподобия, а не частоты. Алгоритм продолжает работу по этому принципу до тех пор, пока не будет достигнут желаемый размер словарного запаса.

Токенизация предложения

Предыдущие алгоритмы токенизации обрабатывали текст как список слов, разделенных пробелами. Это разбиение по пробелам не работает на некоторых языках. В немецком языке составные существительные пишутся без пробелов, например menschenrechte (права человека). Решение состоит в использовании предварительных токенизаторов, зависящих от языка. На немецком языке конвейер NLP использует модуль разделителя составных слов, чтобы проверить, можно ли разделить слово на более мелкие слова. Однако в восточноазиатских языках (например, китайском, японском, корейском и тайском) пробелы между словами не используются. Алгоритм SentencePiece представляет собой простой и независимый от языка токенизатор, предложенный Кудо и др. в 2018 г. и предназначенный для преодоления ограничений, налагаемых отсутствием пробелов. Он воспринимает ввод как необработанный поток, где пробел является частью набора символов. Токенизатор, использующий алгоритм SentencePiece, генерирует символ _, поэтому он встречается в выводе примера модели ALBERT.

Другими популярными языковыми моделями, использующими SentencePiece, являются XLNet, Marian и T5.

Библиотека tokenizers

Библиотека tokenizers предоставляет несколько компонентов, которые позволяют нам построить сквозной токенизатор от предварительной обработки сырого текста до декодирования токенизированных идентификаторов:

Нормализатор→претокенизатор→обучение→постобработка→декодирование

На следующей диаграмме показан конвейер токенизации:

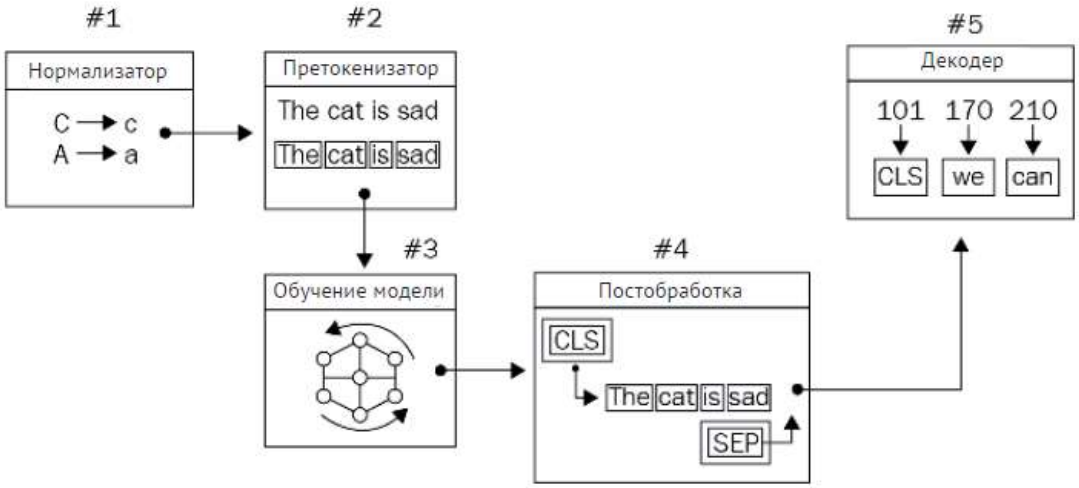

Рис. 3.4. Конвейер токенизации

- Нормализатор позволяет нам применять примитивную обработку текста, такую как перевод в нижний регистр, удаление, нормализация Unicode и удаление акцентов.
- Претокенизатор подготавливает корпус к следующему этапу обучения. Он разбивает ввод на токены в зависимости от правил, таких как пробелы.
- Обучение – это алгоритм токенизации подслов, такой как BPE, BBPE и WordPiece, о которых мы уже говорили. Он исследует подслова / словарный запас и изучает правила генерации.
- Постобработка обеспечивает построение расширенных классов, совместимых с моделями Transformers, такими как BertProcessors. Обычно мы добавляем специальные токены, такие как [CLS] и [SEP], в токенизированный ввод непосредственно перед подачей в модель.
- Декодер отвечает за преобразование идентификаторов токенов обратно в исходную строку и нужен просто для проверки того, что происходит.