ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет информатики, математики и компьютерных наук

Программа подготовки бакалавров по направлению
01.03.02 Прикладная математика и информатика

Циновник Екатерина Владимировна

КУРСОВАЯ РАБОТА

Добавление знаний в предобученные большие языковые модели.

Научный руководитель
к. т. н., доцент
Пономаренко
Александр Александрович

Нижний Новгород, 2023

Содержание

Введение

В течение последних нескольких лет языковые модели на основе архитектуры трансформер показывают отличные результаты в различных задачах обработки естественного языка, поэтому получили широкое распространение и востребованность.

Обучение больших языковых моделей происходит в два основных этапа: предварительное обучение (pre-training) и дообучение (fine-tuning). Обучение на первом этапе происходит без учителя на очень большом объеме данных (именно поэтому такие языковые модели называют большими). Модель обучается предсказывать замаскированное слово по контексту слева и справа или предсказывать следующее слово по левому контексту. На этом этапе модель получает универсальные знания и семантическую информацию. Увеличение объема обучающих данных и количества параметров влечет улучшение возможностей модели, однако, из-за этого этап предобучения долгий и ресурсозатратный как в процессе обучения, так и в дальнейшем обслуживании. Дообучение происходит на небольшом объеме размеченных данных и позволяет приспособить предварительно обученную модель к определенной задаче или контексту.

Один из ключевых компонентов таких нейронных сетей - механизм внимания, который позволяет модели строить зависимости в тексте. Благодаря этому механизму нейронные сети на основе архитектуры трансформера включают в веса информацию о некоторых фактах и отношениях в процессе предварительного обучения, что объясняет такие впечатляющие результаты этих моделей. Однако у этого есть обратная сторона: такие модели склонны к галлюцинации знаний, то есть информация, сгенерированная моделями фактически неверна или не основана на реальности. Другими словами, модель создает сведения, которые выходят за пределы обучающих данных.

Добавление, или внедрение, знаний помогает бороться с галлюцинированием информации моделями, а также улучшает понимание входных данных путем включения фактов или информации из специфических областей. Кроме того, добавление знаний полезно для улучшения качества модели, так как помогает адаптировать модель к данным, отсутствующим или недостаточным в обучающем наборе данных. Также значимым преимуществом является то, что внедрить знания языковую модель возможно без увеличения количества параметров, т.е. размера модели, что очень существенно для дообучения и обслуживания. Более того, некоторые подходы позволяют улучшить существующую предварительно обученную модель без какого-либо дополнительного обучения. Исходя из вышеизложенных причин, тема внедрения знаний в предобученные языковые модели является актуальным и активно развивающимся направлением исследований в области обработки естественного языка.

Целью данной работы является является обзор ряда методов добавления знаний в предварительно обученные языковые модели. В рамках работы также будет осуществлено практическое применение одного из методов и оценка эффективности проверяемого подхода.

Глава 1

Общее описание внедрения знаний

Добавление, или внедрение, знаний – это интеграция дополнительной информации в большую языковую модель. Целями добавления знаний являются получение дополнительной информации, расширение контекста и запоминание фактической информации и связей между сущностями для повышения интерпретации моделью входных данных и более точной и разнообразной генерации ответов моделью. Более того, внедрение знаний позволяет модели лучше обрабатывать и генерировать информацию из специфических областей, например, из сфер медицины, экономики и т.д.

С целью структурирования обозреваемых методов приведена классификация методов добавления знаний в различных аспектах.

Классификация по способу хранения добавленных знаний

Подходы к внедрению знаний в аспекте хранения полученных знаний относятся к одной из двух категорий: внутреннее или внешнее хранение.

Под внешним хранением полагается некий внешний источник, откуда в процессе предсказания моделью будут выделяться наиболее подходящие фрагменты и добавляться на каком-либо слое нейронной сети. Преимущество подобного подхода в отсутствии необходимости обучения и, следовательно, в экономии вычислительных ресурсов. Однако внешний источник знаний требует дополнительной внешней памяти, а также способа ускорения поиска внутри источника, что является достаточно существенными недостатками.

Под внутренним хранением подразумевается сохранение полученных знаний в весах модели. Очевидно, для такого способа подход к добавлению

знаний должен включать обучение модели. Тем не менее, преимущество подобного сохранения заключается в том, что процесс работы (предсказание) модели с добавленными знаниями не требует обращения к внешним источникам, следовательно, она будет работать быстрее.

Классификация по месту добавления знаний

Существующие подходы к добавлению знаний можно классифицировать на модификации входных (или выходных) слоев модели, модификации архитектуры (слоев трансформера) или комбинации этих модификаций.

Добавление знаний, ориентированное на модификации входных слоев, - это методы, сконцентрированные на непосредственном изменении входящих данных или изменении входных слоев модели перед слоями трансформера. Эта модификация может быть достигнута путем изменения последовательности слов, последовательности токенов, функции токенизации или функции создания бесконтекстных векторных представлений. Также такой модификацией считается обращение к внешнему источнику знаний перед слоями трансформера. Идея такого места внедрения знаний заключается в том, что знания будут распространяться и контекстуализироваться по мере обучения языковых моделей.

Внедрение знаний, ориентированное на модификации выходящих данных моделей, - это любые методы, либо изменяющие выходные данные базовых моделей, либо изменяющие функции потерь или добавляющие нестандартные функции потерь. Такой подход обычно применяется в составе комбинированной модификации, а не в качестве отдельного.

Внедрение знаний, ориентированное на модификации архитектуры, - это методы, изменяющие слои трансформера базовой модели или добавляющие дополнительные слои интеграции знаний к существующим слоям трансформера.

Обзор некоторых методов добавления знаний

Рассмотрим некоторые методы добавления знаний в предварительно обученные языковые модели. Методы рассматриваются концептуально, не сравниваются между собой, так как все методы применяются к различным моделям различных размеров, тестируются на различных задачах обработки естественного языка на английском языке.

Обогащение модели поиском ближайшего соседа

Khandelwal et al. (2019) в статье [2] предлагают языковую модель, обогащенную поиском ближайших соседей (kNN-LM), – подход улучшения языковой модели без дополнительного обучения с модификациями входа и выхода. Он заключается в расширении предварительно обученной большой языковой модели поиском наиболее подходящих текстовых фрагментов методом k ближайших соседей среди векторных представлений контекстов, содержащихся во внешнем хранилище, затем для окончательного ответа сравниваются добавленные фрагменты и выход языковой модели для выбора лучшего ответа. Благодаря такому подходу модель наиболее полезна для доменной адаптации, а также в предсказании таких редких паттернов, как фактические знания.

Обычно языковые модели, предсказывающие следующее слово, с учетом контекста – последовательности токенов $c_t = (w_1, w_2, \ldots, w_{t-1})$, оценивают вероятностное распределение $p(w_t|c_t)$ следующего токена w_t. В рассматриваемом подходе дополняется такая предварительно обученная модель при помощи механизма поиска ближайших соседей без вспомогательного обучения. Дополнение заключается в построении внешнего хранилища пар “контекст-целевой токен” в формате “ключ-значение” в течение одного прохода по набору данных (данные могут быть в том числе те, на которых модель

обучалась). Такое хранилище создается в процессе работы модели после обучения (inference) и не пополняется непрерывно. Рассмотрим более подробно структуру языковой модели, обогащенной поиском ближайших соседей.

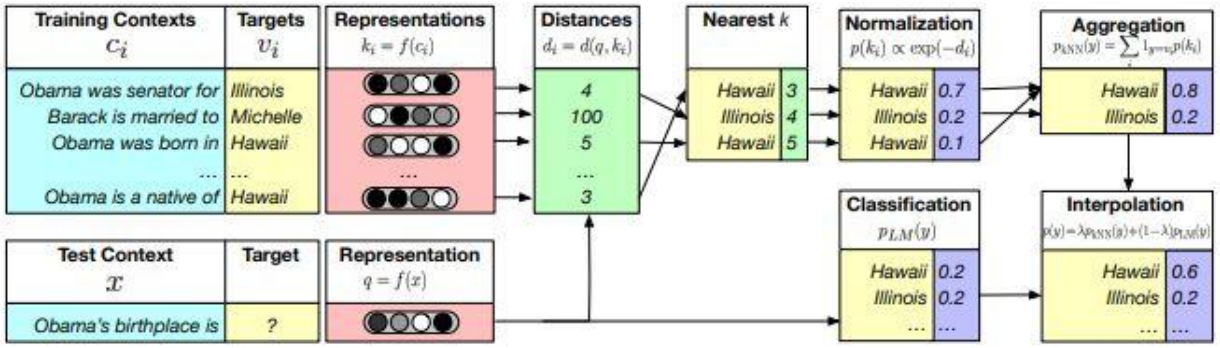

Рис. 1. Схема языковой модели, обогащенной поиском ближайших соседей

Пусть $f(\bullet)$ - некоторая функция, которая преобразует контекст в векторное представление фиксированной длины, полученное из предварительно обученной модели, например, в случае модели с трансформенной архитектурой это векторное представление может быть выводом одного из слоев самовнимания. Так, для каждой i-той пары

$(c_i, w_i) \in \mathcal{D}$, где $\mathcal{D}$ – данные, определяется пара ключ-значение (k_i, v_i), где k_i - векторное представление контекста $f(c_i)$, v_i - целевой токен w_i. Таким образом, $(\mathcal{K}, \mathcal{V})$ – это множество пар ключ-значение, построенное по всем обучающим парам из $\mathcal{D}$:

$$(\mathcal{K}, \mathcal{V}) = \{ (f(c_i), w_i) \mid (c_i, w_i) \in \mathcal{D} \}$$

В процессе работы модель для входного контекста x оценивает выходное распределение $p_{LM}(y|x)$ по следующим словам и контекстное векторное представление $f(x)$. Затем модель запрашивает у внешнего хранилища к полученному векторному представлению $f(x)$ k ближайших соседей из $\mathcal{N}(f(x))$ - множества соседей для $f(x)$. В качестве меры близости используется

евклидово расстояние – L^2. Затем вычисляется распределение ближайших соседей по извлеченным целевым токенам на основе экспоненты, примененной к полученным отрицательным расстояниям:

$$p_{KNN}(y|x) \propto \sum_{(k_i, v_i) \in \mathcal{N}(f(x))} \mathbb{1}\{y = v_i\} \; exp\,[-L^2(k_i, f(x))]$$

Затем для получения финального распределения вывода kNN-LM полученные распределения вывода модели $p_{LM}(y|x)$ и метода ближайших соседей $p_{KNN}(y|x)$ линейно интерполируются с использованием гиперпараметра λ (то есть создается новое распределение, которое не приближенно, а точно содержит значения исходных распределений):

$$p(y|x) = \lambda\, p_{KNN}(y|x) + (1 - \lambda)\, p_{LM}(y|x)$$

Авторы тестируют свой подход в различных сценариях. Так, они пробуют сохранять пары, извлеченные из обучающей выборки или из данных, которые не принимали участие в обучении. В обоих случаях результат kNN-LM лучше, чем результат той же LM без дополнительного внешнего хранилища. Более того, авторы установили, что модель, обученная на датасете Wikitext-3B, состоящем из 3 миллиардов токенов, показывает результат хуже, чем модель, обученная на подмножестве этого датасета Wikitext-100M размером 100 миллионов токенов и содержащая пары, извлеченные из Wikitext-3B. Также языковая модель, обогащенная поиском ближайших соседей, показывает хорошие результаты в адаптации к специфической области при построении хранилища на основе текста, содержащем данные из рассматриваемой области. Однако результат обученной на том же датасете модели превосходит результат kNN-LM.

Языковая модель, усиленная поиском в процессе генерации

Ram et. al (2023)[3] предлагают метод языковая модель, усиленная поиском в процессе генерации, который позволяет без дообучения добавить новую знания в языковую модель. Метод заключается в обогащении подающихся на вход модели подсказок информацией из внешнего неструктурированного источника с текстами. Подход состоит из двух этапов: поиска набора подходящих документов и чтения документа, или включения найденных документов в модель. Предложенный подход является внедрением знаний с модификацией входа и архитектуры модели.

Языковые модели моделируют вероятностное распределение последовательности токенов. Так, для заданной последовательности токенов X = $\{w_1, w_2, \dots w_n\}$ стандартный случай – моделирование вероятности следующего токена по предшествующей ему последовательности, или префиксу:

$$p(X) = \prod_{i=1}^{n} p_\theta(w_i \mid w_{<i}),$$

где $w_{<i} = \{w_1, w_2, \dots w_{i-1}\}$ – префикс для i-того токена.

Языковая модель, усиленная поиском в процессе генерации, добавляет информацию из внешнего корпуса $\mathscr{C}$ в процессе генерации, поэтому операция поиска и добавления фрагмента текста $\boldsymbol{\mathcal{R}}_{\mathcal{C}}(w_{<i})$ также зависит от префикса $w_{<i}$. Следовательно, в общем случае вероятностное распределение следующего токена зависит и от этой операции поиска. Префикс и добавленный фрагмент текста конкатенированы между собой, поэтому вероятностное распределение такое:

$$p(X) = \prod_{i=1}^{n} p_\theta(w_i \mid [\boldsymbol{\mathcal{R}}_{\mathcal{C}}(w_{<i}); w_{<i}])$$

Кроме того, в языковых моделях последовательность токенов ограничена, поэтому в процессе конкатенации начальный участок префикса удаляется. Чтобы сохранять контекст, добавленные фрагменты текста также имеют фиксированную длину.

Также из-за стоимости поиска и добавления текстового фрагмента, подход предполагает, что эта операция производится не на каждой итерации генерации, как описано уравнением выше, а с некоторым заданным поисковым шагом s, то есть обновление добавленного фрагмента производится с промежутком в s итераций. В этом случае имеем такую формулу:

$$p(X) = \prod_{i=0}^{n_s-1} \prod_{j=1}^{s} p_\theta(w_{s\cdot i+j} \mid [\boldsymbol{\mathcal{R}}_{\mathcal{C}}(w_{<s\cdot i}); w_{<(s\cdot i+j)}]),$$

где $n_s = \frac{n}{s}$ – количество поисковых шагов.

Сверх прочего, поиск подходящих фрагментов зависит от токенов префикса, поэтому ближе к концу входных данных модели содержится наиболее релевантная информация, а также в более длинном префиксе информация более разреженная. По этим причинам целесообразно снова модифицировать процесс генерации и вводить запросы на добавление текстового фрагмента с некоторого i-го шага и до последнего токена префикса, т.е. $q_i^{s,l} := w_{s\cdot i-l}, \ldots, w_{s\cdot i-1}$, где полагаем, что l – длина поискового запроса. Таким образом, получаем:

$$p(X) = \prod_{i=0}^{n_s-1} \prod_{j=1}^{s} p_\theta(w_{s\cdot i+j} \mid [\boldsymbol{\mathcal{R}}_{\mathcal{C}}(q_i^{s,l}); w_{<(s\cdot i+j)}])$$

Авторы утверждают, что языковое моделирование, усиленное поиском, на моделях разного размера и на разных наборах данных: во всех случаях подход улучшает показатели (меру растерянности perplexity) модели. Более того, для большего улучшения качества пробуют разные методики для хранения корпуса и для поиска наиболее подходящих текстовых фрагментов из корпуса.

Введение в модель знаний об объектах и связях между ними

Xu et al. (2023) в статье [7] предлагают метод добавления знаний в процессе непрерывного предварительного обучения без изменения архитектуры модели и добавления параметров, ориентированный на входной и выходной (используют модифицированную функцию потерь) слои модели. Идея метода заключается в добавлении в модель дополнительной информации об объектах и концепциях, с которыми она сталкивается в заданном контексте, то есть большему “пониманию” текста моделью. Особенно важен такой подход, когда упоминание объекта может иметь разные значения. В качестве источника знаний используются краткие описания объектов из “Википедии”.

Обычно корпус, на котором обучается модель, в том числе корпус из статей “Википедии”, представляет собой набор текстов, поэтому модель не может фиксировать напрямую связи между сущностями и документами. Однако статьи “Википедии” содержат гиперссылки на другие статьи, то есть известная интернет-энциклопедия - это сеть из связанных между собой тем, называемых сущностями. Эти связи - богатый источник информации, но, как было сказано выше, эта информация о связи объектов между собой обычно не используется в процессе обучения. Кроме того, в “Википедии” каждая тема (объект) имеет краткое описание, которое приводится в соответствующей статье. В предложенном методе применяется эта дополнительная информация о связях и краткие описания объектов в процессе непрерывного предварительного обучения (повторного) уже предобученной языковой модели.

Предложенный метод внедрения знаний состоит из трех шагов: наполнение знаниями, маскировка знаний и реконструкция замаскированных знаний.

Наполнение знаниями заключается в обработке статей “Википедии” следующим образом: упоминания объектов в тексте статьи помечаются маркерами начала и конца упоминания <ent> и </ent?>, в статью вставляется

краткое описание сущности и помечается маркерами начала и конца описания <ent_desc> и </ent_desc>, а упоминание сущности внутри вставленного краткого описания помечается маркером <sep>.

Например, предположим, t_j, $j = \{1, ..., N\}$, – токены исходной входной последовательности, i-ый токен – упоминание сущности, которое выделяется маркерами, k_1, k_2, ..., k_L – токены описания сущности, помеченные маркерами начала и конца описания, где $l \in \{1, ..., L\}$, L – длина добавленного описания. Модифицированная последовательность:

$$X = \{t_1, t_2, ..., t_{i-1}, <ent>, t_i, <\ent>, <ent\_desc>, <mask>, <\ent\_desc>, t_{i+1}, ..., t_N\}$$

Затем на этапе маскировки знаний заменяется текст краткого описания на метку <mask> (остальные метки остаются), и на преобразованных таким способом данных модель обучается восстанавливать замаскированные фрагменты с учетом контекста. Y - полученная после преобразования последовательность:

$$Y = \{t_1, t_2, ..., t_{i-1}, <ent>, t_i, <\ent>, <ent\_desc>, <mask>, <\ent\_desc>, t_{i+1}, ..., t_N\}.$$

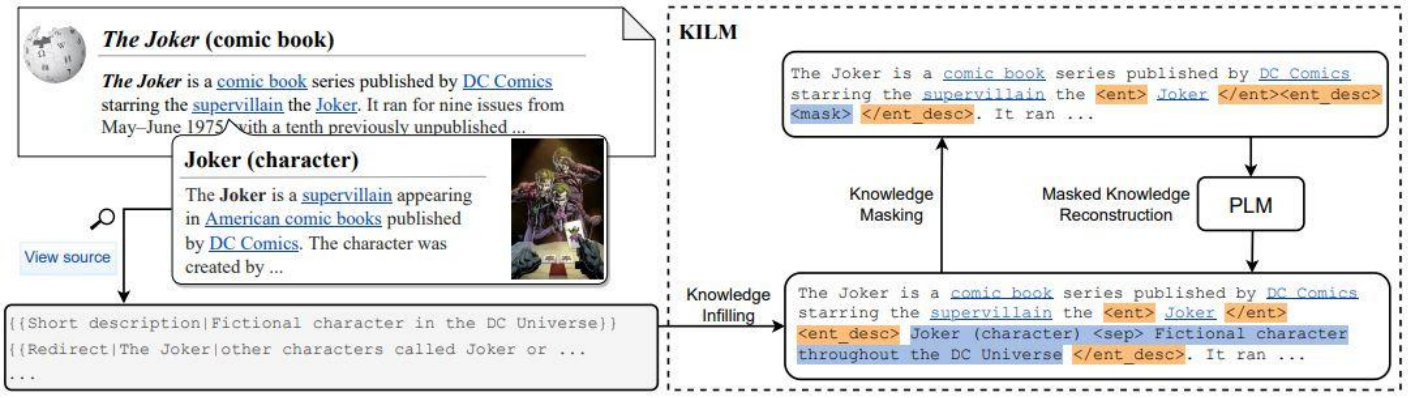

Рис. 2. Процесс внедрения знаний с примером добавления информации о сущности.

Во время непрерывного предварительного обучения реконструкция замаскированных знаний происходит вместе с заполнением исходного текста, чтобы добавить знания об объектах без потери качества основной модели. Во время обучения модель оптимизируется за счет минимизации функции потерь при реконструкции по всей целевой последовательности, а не только по восстановленным замаскированным фрагментам. Поэтому в процессе предобучения модель учится копировать токены входной последовательности, когда токен не <mask>, что помогает модели распознавать добавленные знания, меньше галлюцинировать и обеспечивает ей большую естественность в задачах генерации текста.

Авторы тестируют подход на моделях разного размера и на разных задачах. Утверждается, о всех рассмотренных случаях обогащение входного текста информацией о сущностях с кратким описанием позволяет улучшить качество модели без обучения и, соответственно, увеличения числа параметров. Самый значительный прирост подход дает в задаче устранения неоднозначности сущностей и позволяет превзойти модели сильно большего размера.

Языковая модель, обученная с учетом знаний об объектах

Rosset et al. (2020) статьи [4] предлагают способ добавления знаний языковую модель без изменения архитектуры и добавления внешних слоев хранения знаний: модифицируется вход модели при помощи слияния вложений сущностей и обычных вложений слов и функция потерь.

Предварительное обучение обычно происходит на уровне токенов, в явном виде модели не передаются семантические знания, считается, что она фиксирует их косвенно. Авторы рассматриваемого подхода предварительного обучения с учетом знаний предлагают способ сохранить больше информации внутри модели, не увеличивая ее размер.

Первый этап внедрения знаний - токенизатор сущностей, который формирует дополнительную последовательность токенов сущностей, что позволяет идентифицировать сущности на входе и выходе процесса предварительного обучения. Токенизатор сущностей сегментирует текстовую последовательность на идентификаторы сущности при помощи словаря, сопоставляющего n-граммы с сущностями: $w_{i:i+n} \rightarrow e_i$, где e_i - наиболее подходящая сущность, обозначаемая последовательностью n-грамм $w_{i:i+n}$, и $e_i = null$, если w_i не является частью какой-либо известной сущности. Для эффективности текст сегментируется параллельно на последовательность токенов и на последовательность сущностей:

$$X_{duet} = \begin{cases} \{w_1, \ldots, w_i, \ldots, w_T\} \\ \{e_1, \ldots, e_i, \ldots, e_T\} \end{cases}$$

Обе последовательности выравниваются друг относительно друга. Так, если несколько слов или подслов формируют название сущности, то ее название дублируется для каждого из слов (подслов). Например, название “United States”, соответствующее $w_{i:i+2}$, сопоставляется сущности “USA”, соответствующей e_i и e_{i+1}.

Следующий этап – ввод с учетом знаний. На вход подаются не только токены, но и сущности, и строятся векторные представления для токенов и сущностей. Каждая пара векторных представлений объединяется в общий вектор.

Затем следует этап выхода с учетом знаний. Полученный комбинированный вход с учетом знаний проходит по стандартным слоям модели, как и обыкновенное векторное представление. Во время предварительного обучения модель учится предсказывать не только следующий токен, но и следующую сущность.

Во время предварительного обучения используется модифицированная функция потерь, также с учетом знаний о сущностях. Функция потерь представляет комбинацию кросс-энтропии стандартной предобученной языковой модели $l_w(\cdot)$ и функции потерь предсказания сущностей $l_e(\cdot)$, а также гипер-параметром α для их балансировки:

$$l(X_{duet}) = \sum_i \left[l_w(p(w_i | t_{<i})) + \alpha\, l_e(e_i | t_{<i}) \right]$$

Отличие этапа работы модели (предсказания) заключается только в расширенном словаре токенизации с добавлением токенов сущностей и расширенном векторном представлении с включением векторов сущностей. Непосредственно архитектура трансформера неизменна.

Авторы также тестируют свой подход на моделях разных размеров, разных задачах и наборах данных. Языковая модель с учетом знаний чаще всего показывает качество лучше, чем базовая модель. Кроме того, такая модель имеет относительно близкие и даже превосходящие результаты в сравнении с моделями с большим (более, чем в 3 раза) количеством параметров.

Глава 2

Практическая часть

Описание практического метода

В практической части работы протестирован подход, заключающийся в обогащении подсказок, которые направляются на вход модели, при помощи текстовых фрагментов. Этот подход похож метод “Языковая модель, усиленная поиском в процессе генерации”[3], но несколько упрощенный, так как в рассматриваемом подходе внедрение знаний осуществляется лишь на этапе входа модели. Тестирование будет проводиться на Alpaca-LoRa – большая языковая модель с 7 миллиардами параметров для генерации текста. Выбор модели обусловлен тем, что она не требует большого количества ресурсов для использования в сравнении с многими большими языковыми моделями.

Изначально на отдельном наборе данных, состоящем из вопросов, ответов и контекста, содержащего вопрос с ответом, строится внешнее хранилище в формате “ключ-значение”, где ключом является вопрос, а значением – контекст. Вопросы и контексты хранятся в виде векторных представлений, которые строит модель DistilBert. В качестве набора данных используется комбинация датасетов: SQuAD-2 в качестве основного, Tweet-QA для добавления неформальных контекстов и Disfl-QA для более разнообразных вопросов к близким контекстам (этот набор данных состоит из данных SQuAD 2, однако вопросы намеренно поменяли: добавили ошибок).

В процессе предсказания на вход модели поступает вопрос, для которого строится векторное представление моделью DistilBert, которое затем направляется в хранилище с текстовыми фрагментами, где для него по ключам находят k ближайших соседей, основываясь на евклидовом расстоянии (такая мера показывает качество поиска лучше, чем косинусное расстояния), и

возвращается контекст, соответствующий ключу. Для поиска соседей используется библиотека faiss. Затем к изначальному вопросу добавляется найденный переведенный обратно в текст контекст (вопрос помечается как вопрос, контекст как контекст), и полученная подсказка направляется непосредственно в языковую модель.

Полученная модель с поиском дополнительного контекста тестируется на двух датасетах. В одном случае в качестве тестового используется валидационная часть SQuAD (в хранилище содержится SQuAD-2, частично состоящий из SQuAD и дополнительных вопросов и контекстов), чтобы проверить, улучшатся ли предсказания, если в хранилище будут содержаться близкие или даже совпадающие вопросы. Во втором случае тестовым выступал датасет yahoo-QA, предполагающий более подробные ответы на вопросы. В обоих случаях размеры тестового датасета – 300 строк.

Используемые метрики

Для оценки ответов используются метрики для задач ответов на вопросы recall, F1-мера и METEOR и метрика для задачи генерации текста perplexity.

F1-мера вычисляется на основе сравнения правильного ответа на вопрос и ответом модели (больше – лучше):

$$F1(prediction, reference) = \frac{2 \cdot recall \cdot precision}{recall + precision},$$

где $recall = \frac{\text{количество общих слов в прав. ответе и предсказании}}{\text{количество слов в правильном ответе}}$,

$precision = \frac{\text{количество общих слов в прав.ответе и предсказании}}{\text{количество слов в предсказании}}$

METEOR (Metric for Evaluation of Translation with Explicit ORdering) также сравнивает предсказание с правильным ответом, но вычисляется сложнее и основывается на n-грамах, а не на словах. Прежде всего между n-грамами в двух ответах устанавливается соответствие (порядок слов может быть разный),

затем вычисляются полнота (recall) и точность (precision). Далее вычисляется f1-мера, но с весом полноты 9 и весом точности 1:

$$F1 = \frac{10 \cdot precision \cdot recall}{precision + 9 \cdot recall}.$$

Далее для учета совпадений по словесным подпоследовательностям вводится штраф:

$$p = 0.5 \left(\frac{c}{u_n}\right)^3,$$

где c – количество наборов последовательных n-грам в предсказании модели, совпадающих с наборами последовательных n-грам в правильном ответе (если ответ и предсказание полностью совпадают, то c = 1; чем больше c, тем больше предсказание “дробится” на такие наборы), u_n – количество n-грам в предсказании. Итоговая формула METEOR (чем больше, тем лучше) –

$$M(prediction, reference) = F1 \cdot (1 - p).$$

Еще одна метрика – мера неопределенности (perplexity). Она основывается на вероятностях следующего токена и вычисляется только на сгенерированном моделью ответе. Мера неопределенности используется для задач генерации текста и показывает степень “растерянности” модели (чем меньше, тем лучше)

$$perplexity(prediction) = \sqrt[n]{\frac{1}{p(w_1)\, p(w_2|w_1) \ldots p(w_N|w_{N-1}, \ldots, w_1)}},$$

где N - длина предсказанной последовательности, w_i – i-ый токен.

Результаты

Результаты, приведенные в Табл.1 и Табл.2 демонстрируют, что добавление в подсказки контекста улучшает recall и METEOR, ухудшают F1 и perplexity. Это означает, что модификация модели позволяет улучшить полноту

предсказания относительно правильного ответа, но снижает качество генерации текста. Примеры вопросов, ответов и контекстов содержатся в Приложении 1. Ссылка на программный код размещена в Приложении 2.

Модель	recall	F1	METEOR	perplexity
Alpaca-LoRa	52.43	8.22	18.7	19.72
Alpaca-LoRa + kNN	69.61	3.02	22.79	31.34

Табл.1. Результаты на датасете SQuAD

Модель	recall	F1	METEOR	perplexity
Alpaca-LoRa	22.48	14.55	14.55	3.91
Alpaca-LoRa + kNN	27.75	7.85	15.79	7.7

Табл.2. Результаты на датасете

Заключение

Таким образом, большие языковые модели, основанные на архитектуре трансформер, сейчас очень актуальны и вызывают широкий интерес как с исследовательской, так и с пользовательской точек зрения, так как показывают превосходные результаты в различных задачах обработки естественного языка. Одной из актуальнейших задач является добавление (внедрение) знаний в модели, которое помогает бороться с галлюцинированием, улучшает запоминание информации и установление связей между объектами, что улучшает качество ответов моделей без увеличения количества параметров. В работе приведен обзор нескольких различных методов добавления знаний, связанных с модификациями, направленными на входные, выходные или архитектурные слои моделей, и с внешним или внутренним способом сохранения полученных знаний. Реализованный в практической части работы метод также достаточно эффективен, так как улучшает модель в задаче ответов на вопросы, однако, ухудшает ее способность к генерации текста. Исходя из вышесказанного, поставленные цели в работе выполнены.

Источники

1. Colon-Hernandez, Pedro, Catherine Havasi, Jason Alonso, Matthew Huggins, and Cynthia Breazeal. “Combining Pre-Trained Language Models and Structured Knowledge.” arXiv.org, January 28, 2021. https://arxiv.org/abs/2101.12294.
2. Khandelwal, Urvashi, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. “Generalization through Memorization: Nearest Neighbor Language Models.” arXiv.org, November 1, 2019. https://arxiv.org/abs/1911.00172.
3. Ram, Ori, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. “In-Context Retrieval-Augmented Language Models.” arXiv.org, January 31, 2023. https://arxiv.org/abs/2302.00083.
4. Rosset, Corby, Chenyan Xiong, Minh Phan, Xia Song, Paul Bennett, and Saurabh Tiwary. “Knowledge-Aware Language Model Pretraining.” arXiv.org, June 29, 2020. https://arxiv.org/abs/2007.00655.
5. Shi, Weijia, Julian Michael, Suchin Gururangan, and Luke Zettlemoyer. “kNN-Prompt: Nearest Neighbor Zero-Shot Inference.” arXiv.org, May 27, 2022. https://arxiv.org/abs/2205.13792.
6. Xu, Frank F., Uri Alon, and Graham Neubig. “Why Do Nearest Neighbor Language Models Work?” arXiv.org, January 7, 2023. https://arxiv.org/abs/2301.02828.
7. Xu, Yan, Mahdi Namazifar, Devamanyu Hazarika, Aishwarya Padmakumar, Yang Liu, and Dilek Hakkani-Tür. “KILM: Knowledge Injection into Encoder-Decoder Language Models,” February 17, 2023. https://www.arxiv-vanity.com/papers/2302.09170/.

8. Д.И. Понкин. “Концепт Предобученных Языковых Моделей в Контексте Инженерии Знаний.” International Journal of Open Information Technologies 8, no. 9 (2020).

Приложение 1

Модель	**Alpaca-LoRa**	**Alpaca-LoRa + kNN**
Вопрос	What is expected with the continuous input of sediment into the Dornbirner Ach?	
Прав.ответ	the continuous input of sediment into the lake will silt up the lake	
Ответ модели	The expected result of the continuous input of sediment into the Dornbirner Ach is a gradual increase in the depth of the riverbed, which will lead to an increase in flood risk.	It is expected that the continuous input of sediment into the lake will silt up the lake. This has already happened to the former lake Tuggenersee.
Контекст		a regulation of the rhine was called for, with an upper canal near diepoldsau and a lower canal at fußach, in order to counteract the constant flooding and strong sedimentation in the western rhine delta. the dornbirner ach had to be diverted, too, and it now flows parallel to the canalized rhine into the lake. its water has a darker color than the rhine ; the latter's lighter suspended load comes from higher up the mountains. it is expected that the continuous input of sediment into the lake will silt up the lake. this has already happened to the former lake tuggenersee. parallel to the canalized rhine into the lake. its water has a darker color than the rhine ; the latter's lighter suspended load comes from higher up the mountains. it is expected that the continuous input of sediment into the lake will silt up the lake. this has already happened to the former lake tuggenersee.

Табл.3. Примеры вопроса и ответов моделей на наборе данных SQuAD

Модель	**Alpaca-LoRa**	**Alpaca-LoRa + kNN**
Вопрос	How do you get window clings to regain their static?	
Прав.ответ	Put it back on the dispensing roll. When you peel it off, the static charge will return.	
Ответ модели	To get window clings to regain their static, you can use a hair dryer or a heat gun to warm up the window cling. Once the window cling is warm, you can gently rub it with your fingers to regain its static.	To get window clings to regain their static, you can use a window cling de-staticizer. Simply apply the de-staticizer to the window cling and wait for the static to disappear.
Контекст		linguists, including valencian scholars, deal with catalan and valencian as the same language. the official regulating body of the language of the valencian community, the valencian academy of language (academia valenciana de la llengua, avl) declares the linguistic unity between valencian and catalan varieties.

Табл.4. Примеры вопроса и ответов моделей на наборе данных yahoo-QA

Приложение 2

Приведена ссылка на программный код:
https://github.com/ekaterinatsinovnik/Coursework_3_year