(19) Евразийское патентное ведомство

(11) 045059 (13) B1

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ЕВРАЗИЙСКОМУ ПАТЕНТУ

(45) Дата публикации и выдачи патента
2023.10.27

(21) Номер заявки
202293416

(22) Дата подачи заявки
2022.12.21

(51) Int. Cl. ***G06F 40/00*** (2020.01)
G06F 18/2431 (2023.01)
G06F 40/169 (2020.01)
G06F 40/20 (2020.01)
G06F 40/274 (2020.01)
G06F 40/284 (2020.01)
G06F 40/30 (2020.01)
G06N 20/00 (2019.01)
G10L 15/16 (2006.01)
G10L 15/183 (2013.01)

(54) СПОСОБ И СИСТЕМА КЛАССИФИКАЦИИ ТЕКСТА

(31) **2022114371**
(32) **2022.05.27**
(33) **RU**
(43) **2023.10.25**

(71)(73) Заявитель и патентовладелец:
ПУБЛИЧНОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО "СБЕРБАНК РОССИИ" (ПАО СБЕРБАНК) (RU)

(72) Изобретатель:
Кондюк Никита Евгеньевич, Тихонова Мария Ивановна (RU)

(74) Представитель:
Герасин Б.В. (RU)

(56) US-B1-9959272
US-A1-20190340235
US-A1-20220245354
US-A1-20210209300
US-A1-20200380301

(57) Заявленное изобретение в общем относится к области вычислительной техники, а в частности к способам и системам дообучения языковой модели и решения задачи классификации текста дообученной языковой моделью. Техническим результатом от реализации заявленного решения является обеспечение возможности автоматической генерации подсказки для дообучения языковой модели на решение задачи классификации. В предпочтительном варианте реализации заявлен способ классификации текста языковой моделью, выполняющийся по меньшей мере одним вычислительным устройством и содержащий этапы, на которых получают входной набор данных, соответствующий требуемой задаче классификации, в формате, на основе которого дообучалась языковая модель; выполняют форматирование входного набора данных, дополняя его символами, причем каждый символ соответствует абстрактному псевдо-слову; выполняют токенизацию и векторизацию входного набора данных, причем символы, соответствующие абстрактным псевдо-словам, заменяются на обученные векторные представления символов, сохраненных в файле данных; выполняют обработку данных, полученных на этапе c), дообученной языковой моделью, в ходе которой получают вектор логитов, отражающий вероятностное распределение классов, соответствующих словам словаря языковой модели; выбирают целевые компоненты логитов, соответствующие токенам целевых классов решаемой задачи классификации; определяют из целевых компонентов логитов, полученных на этапе e), компоненту логита, отражающую наибольшее значение вероятности принадлежности к целевому классу; генерируют ответ в текстовой форме, соответствующий выбранной на этапе f) компоненте.

045059 B1

045059 B1

Область техники

Заявленное техническое решение в общем относится к области вычислительной техники, а в частности к способам и системам дообучения языковой модели и решения задачи классификации текста дообученной языковой моделью.

Уровень техники

С развитием информационных технологий неотъемлемой частью множества различных цифровых сервисов являются языковые модели. Так, с помощью языковых моделей решаются задачи поддержания диалога голосовыми помощниками, генерация ответов на вопросы пользователя цифровыми ассистентами, определение эмоциональной окраски текста, формирование аннотаций к статьям и т.д., т.е. прикладные задачи классификации текста. Современные языковые модели содержат в себе достаточное количество знаний (миллиарды просмотренных текстов) о мире, что позволяет решать такими моделями различные задачи путем продолжения входного текста. Основной принцип работы таких моделей заключается в определении (предсказании) вероятности появления следующего слова на основе текстов, на которых указанная модель обучалась (знания модели).

Для решения конкретных задач классификации текста с высокой точностью, в настоящий момент, требуется адаптировать (дообучить) модель в соответствии с решаемой задачей, что является вычислительно сложной и нетривиальной задачей. Так, одним из способов адаптации языковой модели является метод файн-тюнинг (fine-tuning), в ходе которого выполняют полное дообучение модели на новых обучающих данных, соответствующих решаемой задаче. Недостатками указанного способа является вычислительная ресурсоёмкость из-за обучения всех весов; увеличение требуемой памяти ввиду хранения копии дообученной модели по размеру равной исходной модели; высокие требования к размеру обучающего датасета.

Также из уровня техники известны способы дообучения языковой модели, для решения требуемой задачи классификации, few-shot и zero-shot. Указанные способы основаны на подборе входного текста (подсказки), который модель будет продолжать для получения ответа на решаемую задачу классификации. Однако подбор правильной подсказки осуществляется вручную и является сложной и нетривиальной задачей.

Из уровня техники также известно решение, раскрытое в заявке на патент США № 2021/0280167 A1 (SMITH MARIA E. [US] et al.), опубл. 09.09.2021. Указанное решение, в частности, раскрывает возможность генерирования подсказки с помощью аудиоввода для корректного синтезирования речи с приданием указанной речи требуемых лингвистических характеристик.

Недостатками данного решения являются невозможность решения задач классификации текста ввиду особенности решения, невозможность автоматического подбора подсказки для решения задач классификации. Кроме того, указанное решение не обладает универсальностью.

Общим недостатком существующих решений является отсутствие эффективного и точного способа автоматической адаптации языковой модели для решения требуемых задач классификации текста. Также указанное решение должно обеспечивать возможность адаптации языковой модели на малых объемах обучающих данных с сохранением высокой точности для решаемой задачи, что, как следствие, снижает необходимый объем и требования к вычислительным ресурсам, а также объем памяти, требуемый для хранения дообученной языковой модели. Кроме того, такого рода решение должно обеспечивать универсальность технологии адаптации модели, позволяющей генерировать произвольные форматы подсказок, что обеспечивает поддержку произвольной структуры входных данных.

Раскрытие изобретения

Данное техническое решение направлено на устранение недостатков, присущих существующим решениям, известным из уровня техники. В заявленном техническом решении предлагается новый подход к дообучению языковой модели для классификации текста.

Таким образом, решается техническая проблема обеспечения возможности автоматической генерации подсказки для адаптации языковой модели для классификации текста.

Техническим результатом, проявляющимся при решении вышеуказанной проблемы, является обеспечение возможности автоматической генерации подсказки для дообучения языковой модели на решение задачи классификации.

Дополнительным техническим результатом, достигающимся при решении данной проблемы, является снижение вычислительной мощности на обучение языковой модели и объема памяти для хранения языковой модели для классификации текста.

Указанные технические результаты достигаются благодаря осуществлению способа автоматического дообучения языковой модели для классификации текста, выполняющегося по меньшей мере одним вычислительным устройством и содержащего этапы, на которых:

a) принимают языковую модель и обучающий входной набор данных, содержащий по меньшей мере список полей входного набора данных;

b) выбирают формат подсказки, на основе списка полей обучающего входного набора данных, для заданной задачи классификации и выполняют форматирование обучающего входного набора данных, на основе выбранного формата подсказки, в ходе которого выполняют дополнение упомянутого набора

данных символами, при этом каждый символ соответствует абстрактному псевдо-слову;

c) выполняют токенизацию и векторизацию отформатированного на этапе b) обучающего входного набора данных, причем векторизация выполняется только для токенов, которые не относятся к символам;

d) инициализируют обучаемые векторные представления для токенизированных символов, соответствующих абстрактным псевдо-словам, и заменяют упомянутые токенизированные символы на инициализированные обучаемые векторные представления символов;

e) подают на вход языковой модели отформатированные обучающий входной набор данных в виде векторных представлений, и получают вероятности аналогов целевых классов;

f) обновляют обучаемые векторные представления, на основе истинных значений целевых классов для каждого обучающего объекта, итеративно повторяя этапы e)-f) до выполнения критерия останова;

g) формируют файл данных и сохраняют обученные векторные представления символов, соответствующие абстрактным псевдо-словам.

В одном из частных вариантов реализации способа форматирование обучающего входного набора данных выполняется с использованием шаблона для форматирования.

В другом частном варианте реализации способа шаблон для форматирования выбирается на основе решаемой задачи классификации.

В другом частном варианте реализации способа задача классификации представляет собой по меньшей мере:

i) бинарные вопросно-ответные системы;

ii) бинарная классификация на распознавание причинно-следственной связи между двумя предложениями;

iii) бинарная классификация на выбор одной из двух альтернатив;

iv) задача на машинное чтение в форме бинарной классификации;

v) задача бинарной классификации на распознавание причинно-следственных связей между посылкой и гипотезой.

Кроме того, заявленные технические результаты достигаются за счет способа классификации текста языковой моделью, дообученной согласно этапам способа автоматического дообучения языковой модели для классификации текста, выполняющегося по меньшей мере одним вычислительным устройством и содержащего этапы, на которых:

a) получают входной набор данных, соответствующий требуемой задаче классификации, в формате, на основе которого дообучалась языковая модель;

b) выполняют форматирование входного набора данных, дополняя его символами, причем каждый символ соответствует абстрактному псевдо-слову;

c) выполняют токенизацию и векторизацию входного набора данных, причем символы, соответствующие абстрактным псевдо-словам, заменяются на обученные векторные представления символов, сохраненные в файле данных;

d) выполняют обработку данных, полученных на этапе c), дообученной языковой моделью, в ходе которой получают вектор логитов, отражающий вероятностное распределение классов, соответствующих словам словаря языковой модели;

e) выбирают целевые компоненты логитов, соответствующие токенам целевых классов решаемой задачи классификации;

f) определяют из целевых компонентов логитов, полученных на этапе e), компоненту логита, отражающую наибольшее значение вероятности принадлежности к целевому классу;

g) генерируют ответ в текстовой форме, соответствующий выбранной на этапе f) компоненте.

Кроме того, заявленные технические результаты достигаются за счет системы автоматического дообучения языковой модели для классификации текста, содержащей:

по меньшей мере один процессор;

по меньшей мере одну память, соединенную с процессором, которая содержит машиночитаемые инструкции, которые при их выполнении по меньшей мере одним процессором обеспечивают выполнение способа автоматического дообучения языковой модели для классификации текста.

Кроме того, заявленные технические результаты достигаются за счет системы классификации текста дообученной языковой моделью, содержащей:

по меньшей мере один процессор;

по меньшей мере одну память, соединенную с процессором, которая содержит машиночитаемые инструкции, которые при их выполнении по меньшей мере одним процессором обеспечивают выполнение способа классификации текста языковой моделью.

Краткое описание чертежей

Признаки и преимущества настоящего изобретения станут очевидными из приводимого ниже подробного описания изобретения и прилагаемых чертежей.

Фиг. 1 иллюстрирует блок-схему общего вида заявленной системы автоматического дообучения языковой модели для классификации текста.

Фиг. 2 иллюстрирует блок-схему общего вида заявленной системы классификации текста языковой моделью.

Фиг. 3 иллюстрирует блок-схему выполнения способа автоматического дообучения языковой модели для классификации текста.

Фиг. 4 иллюстрирует блок-схему выполнения способа классификации текста языковой моделью.

Фиг. 5 иллюстрирует пример общего вида вычислительной системы, которое обеспечивает реализацию заявленного решения.

Осуществление изобретения

Ниже будут описаны понятия и термины, необходимые для понимания данного технического решения.

Модель в машинном обучении (МО) - совокупность методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач.

Векторное представление слов (word embeddings, эмбеддинги) - общее название для различных подходов к моделированию языка и обучению представлений в обработке естественного языка, направленных на сопоставление словам (и, возможно, фразам) из некоторого словаря векторов из n-мерного вещественного пространства R_n.

Токенизация - это процесс разбиения текста на текстовые единицы или токены (чаще всего в качестве таких единиц выступают слова, но это могут быть также буквы, части предложения, сочетания слов и т.д.).

Языковая модель - это вероятностное распределение на множестве словарных последовательностей. В данном патенте термин "языковая модель" употребляется для описания нейросетевых языковых моделей, которые выполнены с возможностью моделирования языка посредством оценки вероятности той или иной последовательности символов.

Логиты - в данном патенте термин употребляется для обозначения выходов с последнего слоя нейросетевой языковой модели, которые представляют собой логарифм отношения шансов. Понятие логитов тесно связано с вероятностным распределением, и, как следствие, вектор логитов можно преобразовать в вероятностное распределение.

Подсказка (prompt) - это произвольный входной текст и/или последовательность символов и/или соответствующая им последовательность входных эмбеддингов, встраиваемая в обрабатываемый языковой моделью текст и обеспечивающая изменение результата обработки текста языковой моделью. Более подробно термин подсказка раскрыт в уровне техники, см., например, источник, найдено в Интернет по ссылке: https://arxiv.org/pdf/2005.14165.pdf.

Заявленное техническое решение предлагает новый подход, обеспечивающий возможность автоматической генерации подсказки для языковой модели, что позволяет эффективно дообучить языковую модель на классификацию текста. Такой подход обеспечивает снижение затрачиваемых на обучение языковой модели классификации текста требуемых вычислительных ресурсов, сохраняя при этом высокую точность классификации. Кроме того, за счет исключения необходимости хранения полностью дообученной, в соответствии с известным уровнем техники, языковой модели сокращается требуемый объем памяти (на несколько порядков).

Заявленное техническое решение может быть реализовано на компьютере, в виде автоматизированной информационной системы (АИС) или машиночитаемого носителя, содержащего инструкции для выполнения заявленных предпочтительных вариантов реализации решения.

Техническое решение также может быть реализовано в виде распределенной компьютерной системы или вычислительного устройства.

В данном решении под системой подразумевается компьютерная система, ЭВМ (электронно-вычислительная машина), ЧПУ (числовое программное управление), ПЛК (программируемый логический контроллер), компьютеризированные системы управления и любые другие устройства, способные выполнять заданную, чётко определённую последовательность вычислительных операций (действий, инструкций). Под устройством обработки команд подразумевается электронный блок либо интегральная схема (микропроцессор), исполняющая машинные инструкции (программы). Устройство обработки команд считывает и выполняет машинные инструкции (программы) с одного или более устройства хранения данных, например, таких устройств, как оперативно запоминающие устройства (ОЗУ) и/или постоянные запоминающие устройства (ПЗУ). В качестве ПЗУ могут выступать, но не ограничиваясь, жесткие диски (HDD), флэш-память, твердотельные накопители (SSD), оптические носители данных (CD, DVD, BD, MD и т.п.) и др.

Программа - последовательность инструкций, предназначенных для исполнения устройством управления вычислительной машины или устройством обработки команд.

Термин "инструкции", используемый в этом документе, может относиться, в общем, к программным инструкциям или программным командам, которые написаны на заданном языке программирования для осуществления конкретной функции, такой как, например, кодирование и декодирование текстов, фильтрация, ранжирование, трансляция текстов в диалоговую систему и т.п. Инструкции могут быть

осуществлены множеством способов, включающих в себя, например, объектно-ориентированные методы. Например, инструкции могут быть реализованы посредством языка программирования Python, C++, Java, Python, различных библиотек (например, MFC; Microsoft Foundation Classes) и т.д. Инструкции, осуществляющие процессы, описанные в этом решении, могут передаваться как по проводным, так и по беспроводным каналам передачи данных, например Wi-Fi, Bluetooth, USB, WLAN, LAN и т.п.

Поскольку изначально языковая модель не дообучена на классификацию текста, а способна только выдавать вероятности предсказания следующего слова для входного текста, то для решения задач классификации необходимо выполнить дообучение языковой модели. Как понимается в данном техническом решении, дообучение языковой модели представляет собой адаптацию языковой модели к обеспечению возможности классификации текста. Так, в настоящем варианте реализации заявленного решения дообучение языковой модели на классификацию текста осуществляется с помощью генерации подсказки (prompt) для указанной модели. Сама же классификация текста осуществляется с помощью обработки языковой моделью входных данных, дополненных и поданных в определенном виде (объединенных с подсказкой). Указанная подсказка обеспечивает точную обработку данных языковой моделью и соответственно точный и качественный результат классификации, однако, как указывалось в уровне техники, генерация подсказки является сложной и трудоемкой задачей. Аспекты настоящего технического решения направлены на решение указанной проблемы с помощью автоматической генерации подсказки и более подробно описано ниже. На фиг. 1 приведен общий вид системы 100 автоматического дообучения языковой модели для классификации текста. В первом предпочтительном варианте осуществления система 100 включает в себя основные функциональные элементы, такие как модуль получения данных 101, модуль хранения языковой модели 102, модуль форматирования 103, модуль вербализации целевых классов 104, модуль предобработки 105, модуль параметризации обучаемых эмбеддингов 106, модуль инъекции обучаемых эмбеддингов 107, модуль обучения 108. Более подробно элементы системы 100 раскрыты на фиг. 5.

Модуль получения данных 101 может быть реализован на базе по меньшей мере одного вычислительного устройства, например на базе элементов системы 500, раскрытой на фиг. 5, и выполнен с возможностью получения языковой модели и обучающего входного набора данных, содержащего по меньшей мере список полей входного набора данных.

Так, языковой моделью, в частном варианте осуществления изобретения, может являться адаптируемая предобученная языковая модель, например BERT, GPT3, ELMo, Transformer и т.д. В одном частном варианте осуществления языковая модель выбирается типа трансформер-декодер. В предпочтительном варианте реализации была использована языковая модель RuGPT3-Large, доступная по ссылке, найдено в Интернет: https://huggingface.co/sberbank-ai/rugpt3large_based_on_gpt2. Основной принцип работы языковой модели заключается в том, что на вход модели подается текстовая последовательность, а модель, на основе исторических данных, предсказывает вероятность появления следующего слова. Кроме того, в модели учитывается, какие комбинации слов и в каком порядке чаще всего встречаются в языке вместе. И чем больше и разнообразнее набор текстов, на которых она обучается, тем более качественные зависимости можно улавливать моделью и воспроизводить их на новых данных.

Обучающим набором данных является набор данных, содержащий в себе список полей, определяющих решаемую задачу классификации. В качестве задачи классификации может выступать, например, бинарный ответ в вопросно-ответных системах; бинарная классификация на распознавание причинно-следственной связи между двумя предложениями; бинарная классификация на выбор одной из двух альтернатив; задача на машинное чтение в форме бинарной классификации; задача бинарной классификации на распознавание причинно-следственных связей между посылкой и гипотезой и т.д. Результатом решения такого рода задач является ответ языковой модели. Таким образом, обучающий набор данных содержит в себе список полей, которые определяют формат решаемой задачи классификации. Так, например, для решения логической задачи определения причинно-следственной связи в виде предоставления бинарного ответа обучающий набор данных будет содержать следующие поля, например: "текст 1:" "Вася ловит рыбу", "текст 2:" "Вася дома?", "ответ:" нет. Стоит отметить, что список полей зависит от задачи классификации. Так, например, для бинарной вопросно-ответной системы список полей входного набора данных будет содержать следующие поля: "текст:", "вопрос:", "ответ:". Соответственно для каждого типа решаемой задачи подбирается свой обучающий входной набор данных, на основе которого генерируется подсказка. Так, в качестве примера приведем размеры и структуру (формат) обучающих наборов для нескольких задач классификации текста. Для задачи бинарного ответа на вопрос по тексту (бинарная вопросно-ответная система) был использован размер обучающего набора данных в размере 1749 примеров, каждый из которых был представлен в формате "текст:", "вопрос:", "ответ:". Для задачи бинарной классификации на распознавание причинно-следственной связи между двумя предложениями (задача определения, учитывая два текстовых фрагмента, выводится ли (может быть выведено) значение одного текста из другого текста) размер обучающего набора составлял 2616 примеров, каждый из которых был представлен следующим списком полей: "полный текст:", "вывод:", "ответ:".

Указанный обучающий набор далее подается в модуль форматирования 103. Модуль хранения языковой модели 102 может быть реализован на базе по меньшей мере одного вычислительного устройства

и предназначен для хранения исходной языковой модели. Так, модуль 102 может представлять собой, например, постоянное запоминающее устройства (ПЗУ), которое может являться одним или более средств для постоянного хранения данных, например жесткий диск (HDD), твердотельный накопитель данных (SSD), флэш-память (EEPROM, NAND и т.п.), оптические носители информации (CD-R/RW, DVD-R/RW, BlueRay Disc, MD) и др.

Модуль форматирования 103 может быть реализован на базе по меньшей мере одного вычислительного устройства, оснащенного соответствующим программным обеспечением, и предназначен для определения формата подсказки, на основе списка полей обучающего входного набора данных, для заданной задачи классификации и выполнения форматирования обучающего набора данных, на основе определенного формата подсказки, в ходе которого выполняют дополнение упомянутого набора данных символами, при этом каждый символ соответствует абстрактному псевдо-слову.

Общий принцип работы модуля 103 заключается в преобразовании обучающего набора данных в строку, составляя её из списка полей, который содержится в указанном наборе данных, и добавления символов между полями обучающего набора, причем количество добавляемых символов зависит от решаемой задачи классификации. Таким образом, при форматировании обучающего набора сначала обучающий набор преобразовывается в строку и далее дополняется символами, соответствующими абстрактным псевдо-словам. Так, например, символ {P} является абстрактными псевдо-словами, а фрагмент {имя_поля} является полем обучающего набора, содержимое которого необходимо включить на данную позицию при формировании подсказки к конкретному примеру.

Так, например, в одном частном варианте осуществления результатом работы модуля 103 при форматировании обучающего входного набора данных для определения наличия причинно-следственной связи будет следующий формат: "{P}{P}{P}{текст 1}{P}{P}{P}{текст 2}{P}{P}{P}". Где символ {P} соответствует абстрактному псевдо-слову, т.е. для данного символа не существует человеко-читаемого текстового аналога. Так, символ ({P}) соответствует абстрактному символу, для которого будет меняться его векторное представление (обучение векторного представления), не соответствующее никакому реальному слову (человеку-читаемому аналогу) в словаре языковой модели. Стоит отметить, что указанный частный вариант приведен для решаемой задачи наличия причинно-следственной связи между двумя фрагментами текста. Так, в еще одном частном варианте осуществления формат подсказки может определяться на основе шаблона для форматирования, который, в свою очередь, зависит от решаемой задачи классификации. Так, в еще одном частном варианте осуществления в качестве формата подсказки, на основе которого форматируется обучающий входной набор данных, шаблон затравки может быть выражен в виде функции, соответствующей решаемой задачи классификации, например <P>{текст}<P>{вопрос}<P>, с помощью которой происходит форматирование входных полей обучающего входного набора данных. Кроме того, определение выбираемого шаблона для форматирования может быть получено на основе типа списка полей из входного обучающего набора данных, например, посредством анализа названия полей в указанном наборе, например, с помощью регулярных выражений. Возвращаясь к указанному примеру, при определении названия входных полей как {текст}, {вопрос}, будет применен шаблон для бинарной вопросно-ответной системы и т.д.

Модуль вербализации целевых классов 104 может быть реализован на базе по меньшей мере одного вычислительного устройства, оснащенного соответствующим программным обеспечением. Указанный модуль является опциональным и предназначен для обработки целевых классов обучающего набора данных в случае несоответствия их формата требуемому. Так, при наличии в обучающем входном наборе данных целевых классов (поле "ответ"), изложенных не в текстовом виде, указанный модуль 104 выполнен с возможностью сопоставления возможным в заданной задаче классификации целевым классам, содержащимся в обучающем входном наборе данных, их текстовые аналоги. Так, указанный модуль 104 преобразовывает множество целевых классов в текстовую форму. Данный модуль ставит в соответствие каждому целевому классу три сущности: слово, соответствующее понятию класса в естественной речи (например, "да" для положительного класса и "нет" для отрицательного), индекс данного слова в словаре входных эмбеддингов (векторных представлений), порядковый индекс класса. Стоит отметить, что целевыми классами в решаемой задачи является поле "ответ".

Например, продолжая описанный пример, целевыми классами которого являются true/false (поле "ответ" в обучающем наборе данных), будет присвоено следующее: true/false => "да"/"нет" => 349/1386, где последние цифры соответствуют индексам слов да и нет в словаре языковой модели.

Модуль предобработки 105 может быть реализован на базе по меньшей мере одного вычислительного устройства, оснащенного соответствующим программным обеспечением, и включать набор моделей для токенизации текста и векторизации токенизированного текста, например одну или несколько моделей машинного обучения для преобразования текстовой информации в векторную форму, например, BERT, ELMo, ULMFit, XLNet, RoBerta, RuGPT3 и др. В одном частном варианте осуществления модуль 105 может быть реализован на базе элементов системы 500, которая более подробно раскрыта на фиг. 5. Стоит отметить, что определенный метод токенизации и векторизации зависит от языковой модели, используемой в заявленном решении и хранящейся в модуле 102. Например, при использовании модели RuGPT3 токенизация осуществляется методом BPE (Byte Pair Encoding), а последующая векториза-

ция - путем замены каждого токена на его индекс в словаре языковой модели, составленном на этапе изначального обучения модели. Кроме того, в еще одном частном варианте осуществления в качестве метода токенизации может использоваться токенизация по словам. Пример токенизации по словам и векторизации слов индексами в словаре: 'мама мыла раму' → ['мама', 'мыла', 'раму'] → [235, 376, 1056].

Кроме того, в еще одном частном варианте осуществления может выполняться векторизация каждого токена, полученного в процессе токенизации, например, с помощью прямого кодирования (one hot encoding). Так, например, при токенизации на основе алгоритма BPE каждый токен, полученный в ходе указанного процесса токенизации, представлен в словаре своим индексом, отображающим позицию в указанном словаре. Таким образом, каждый токен представляет бинарный вектор (значения 0 или 1), а единица ставится тому элементу, который соответствует номеру токена в словаре, что позволяет представить каждый токен в виде вектора фиксированной длины, соответствующей размерности словаря (например, размерности 3000 на 1). Для специалиста в данной области техники будет очевидно, что для векторизации токенов можно применять и другие алгоритмы векторизации, например алгоритмы Word2vec и т.д., не ограничиваясь. Стоит отметить, что для токенизации символов, добавленных модулем 103, в одном частном варианте осуществления в токенизатор может добавляться соответствующий токен, на который указанные символы будут заменяться. Это необходимо ввиду того, что в исходном токенизаторе может отсутствовать такое слово, как добавленный символ, что приведет к некорректной токенизации текста. Кроме того, в еще одном частном варианте осуществления присваиваемый токен может быть получен вместе с обучающим входным набором данных.

Модуль параметризации обучаемых эмбеддингов 106 может быть реализован на базе по меньшей мере одного вычислительного устройства, оснащенного соответствующим программным обеспечением, и включать набор моделей для векторизации токенизированного текста, например, описанных в модуле 105. Указанный модуль 106 выполнен с возможностью инициализации обучаемых векторных представлений для токенизированных символов, соответствующих абстрактным псевдо-словам, и замены упомянутых токенизированных символов на инициализированные обучаемые векторные представления символов. Под обучаемым векторным представлением (обучаемый эмбеддинг) понимается векторное представление, выполненное с возможностью динамического (итеративного) обновления для приближения к эталонным результатам решаемой задачи классификации, выдаваемых языковой моделью (обучаемые векторные представления - итеративно обновляемые в процессе обучения, с целью повышения качества решаемой задачи, вектора). Т.е. значения обучаемых векторных представлений не зафиксированы и могут обновляться в соответствии с выходным слоем языковой модели. Более подробно указанный термин раскрыт по ссылке, найдено в Интернет: http://www.generalized.ru/Attention_Is_All_You_Need. Принцип работы модуля 106 схож с принципом работы модуля 105 и заключается в векторизации токенизированного текста (символов), однако содержит дополнительный этап инициализации значения эмбеддингов для указанных символов ввиду отсутствия векторного представления в стандартных средствах векторизации для псевдо-слов.

Модуль инъекции обучаемых эмбеддингов 107 используется для объединения последовательности эмбеддингов, порожденных модулем 106, с эмбеддингами токенов, полученных после токенизации обучающего набора модулем 105. При этом два множества эмбеддингов объединяются с учетом порядка, в котором они расположены после форматирования модулем 103. Таким образом, в результате работы модуля 107 все токены, отличные от токенов, соответствующие псевдо-словам, заменяются на эмбеддинги, полученные в результате векторизации обучающего набора данных, а токены, соответствующие псевдо-словам, последовательно заменяются на обучаемые эмбеддинги, порожденные модулем 106.

Модуль обучения 108. Указанный модуль 108 может представлять собой вычислительное устройство, выполняющее алгоритм градиентного спуска с помощью произвольного алгоритма оптимизации (например, стохастического градиентного спуска, Adam, AdamW), где градиенты вычисляются по алгоритму обратного распространения ошибки с целью обновления обучаемых векторных представлений. Указанный модуль 108 выполнен с возможностью осуществлять подбор оптимальных значений для обучаемых эмбеддингов, например, с помощью стохастического градиентного спуска. Более подробно указанные алгоритмы раскрыты в уровне техники, например, по ссылке: https://ru.wikipedia.org/wiki/%D0%93%Dl%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%82%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D1%83%D1%81%D0%BA,
https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%82%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D1%83%D1%81%D0%BA,
https://habr.com/ru/post/318970 /. Кратко опишем этот процесс.

Список входных эмбеддингов с модуля 107 подается в слои языковой модели, которая на выходе выдает распределение вероятностей следующего токена. На основе целевых классов, определенных модулем 104, выбираются только вероятности, соответствующие целевым классам, и вычисляется функция потерь между выбранными вероятностями и индексом целевого класса. После чего с помощью процедуры обратного распространения ошибки вычисляются градиенты для обучаемых эмбеддингов и выполняется их обновление. В ходе обучения обучаемые эмбеддинги будут меняться таким образом, чтобы математическое ожидание значения функции потерь на любом объекте из распределения, соответствующе-

го распределению объектов обучающего датасета, уменьшалось.

Для специалиста в данной области техники очевидно, что, хотя и описанные выше модули представлены как отдельные устройства, указанные модули также могут быть объединены в составе одной системы, например системы 500.

Далее рассмотрим общий вид системы классификации текста дообученной языковой моделью.

На фиг. 2 приведен общий вид системы 200 классификации текста языковой моделью. В предпочтительном варианте осуществления система 200 включает в себя основные функциональные элементы, такие как модуль получения данных 201, модуль хранения языковой модели 202, модуль предобработки 203, модуль обработки 204, модуль постобработки 205. Более подробно элементы системы 200 раскрыты на фиг. 5.

Модуль получения данных 201 выполнен с возможностью получения входного набора данных, соответствующего требуемой задаче классификации, в формате, на основе которого дообучалась языковая модель, и дообученную языковую модель.

Входной набор данных, соответствующий требуемой задаче классификации, в формате, на основе которого дообучалась языковая модель, представляет собой набор данных, который структурно совпадает с обучающим набором данных (содержит аналогичные поля), однако в нем отсутствует результат решаемой задачи классификации.

Так, указанный входной набор данных может быть получен, например, от диалоговой системы посредством канала связи, например сети Интернет, и может содержать, например, для решаемой задачи классификации текста для ответа на бинарный вопрос, набор, состоящий из двух полей: "текст:", "вопрос:". Т.е. входной набор представляет собой непосредственный текст, который до этого не был подан в языковую модель, в формате, схожем по структуре с обучающим входным набором данных.

Дообученная языковая модель может быть получена из модуля 202. Указанная дообученная языковая модель является языковой моделью, полученной в результате работы системы 100. Так, дообученная языковая модель может представлять языковую модель и подсказку с обученными, для решения задачи классификации, векторными представлениями (со значениями обученных векторных представлений символов). В одном частном варианте осуществления дообученная языковая модель может быть получена непосредственно от модуля 102.

Модуль хранения языковой модели 202 может представлять собой, например, постоянное запоминающее устройства (ПЗУ), которое может являться одним или более средств для постоянного хранения данных, например жесткий диск (HDD), твердотельный накопитель данных (SSD), флэш-память (EEPROM, NAND и т.п.), оптические носители информации (CD-R/RW, DVD-R/RW, BlueRay Disc, MD) и др. Указанный модуль предназначен для хранения языковой модели, дообученной в соответствии с работой системы 100.

Модуль предобработки 203 может быть реализован на базе по меньшей мере одного вычислительного устройства, оснащенного соответствующим программным обеспечением, и предназначен для форматирования входного набора данных, дополняя его символами и токенизации, и векторизации отформатированного входного набора данных. Так, указанный модуль 203 в одном частном варианте осуществления может содержать в себе модули 103-105. В еще одном частном варианте осуществления модуль 203 содержит в себе средства, более подробно раскрытые на фиг. 5, обеспечивающие выполнение предписанных функций. Так, на вход модуля 203 поступает входной набор данных. На первом этапе работы модуль 203 выполнен с возможностью форматирования входного набора данных в соответствии с типом решаемой задачи классификации. Как было указано ранее, для разных задач классификации могут быть сгенерированные разные подсказки под соответствующий тип решаемой задачи. Так, в одном частном варианте осуществления форматирование входного набора данных осуществляется на основе шаблона для форматирования. Стоит отметить, что тип решаемой задачи, а следовательно, и тип шаблона форматирования могут поступать, например, совместно с входным набором данных. Кроме того, в еще одном частном варианте осуществления форматы шаблонов для различных задач могут храниться в памяти системы 200, например, в виде файла и могут выбираться в соответствии с типом входных полей, определяемых, например, с помощью регулярных выражений.

Общий принцип форматирования заключается в преобразовании входного набора данных в строку, составляя её из списка полей, который содержится в указанном наборе данных, и добавлении символов между полями входного набора, причем количество и расположение символов в процессе форматирования соответствуют формату подсказки. Таким образом, при форматировании входного набора сначала входной набор преобразовывается в строку и далее дополняется символами, причем количество и позиция указанных символов соответствуют количеству и позициям символов, добавленных в процессе форматирования обучающего набора модулем 103 для конкретной задачи классификации. Так, например, в одном частном варианте осуществления результатом форматирования входного набора будет следующий формат: "{P}{P}{P}{текст 1}{P}{P}{P}{текст 2}{P}{P}{P}". Причем указанный формат соответствует формату, выбранному в процессе работы системы 100 для задачи классификации. После форматирования указанного набора в соответствии с форматом решаемой задачи модуль 203 выполняет токенизацию и векторизацию входного набора данных.

Токенизация и векторизация могут выполняться с помощью набора моделей для токенизации текста и векторизации токенизированного текста, например одну или несколько моделей машинного обучения для преобразования текстовой информации в векторную форму, например BERT, ELMo, ULMFit, XLNet, RoBerta, RuGPT3 и др. Кроме того, в еще одном частном варианте осуществления может выполняться векторизация каждого токена, полученного в процессе токенизации, например, с помощью прямого кодирования (one hot encoding). Так, например, при токенизации на основе алгоритма BPE каждый токен, полученный в ходе указанного процесса токенизации, представлен в словаре своим индексом, отображающим позицию в указанном словаре. Таким образом, каждый токен представляет бинарный вектор (значения 0 или 1), а единица ставится тому элементу, который соответствует номеру токена в словаре, что позволяет представить каждый токен в виде вектора фиксированной длины, соответствующей размерности словаря (например, размерности 3000 на 1). Для специалиста в данной области техники будет очевидно, что для векторизации токенов могут применять и другие алгоритмы векторизации, например алгоритмы Word2vec и т.д., не ограничиваясь. Стоит отметить, что алгоритм векторизации будет зависеть от конкретной языковой модели, применяемой для реализации данного технического решения.

Стоит отметить, что значения для токенизации и последующей векторизации символов получают из файла данных, хранящего обученные эмбеддинги. Так, в одном частном варианте осуществления, как упоминалось выше, файл с разными типами сгенерированных подсказок могут храниться и/или поступать в систему 200. Так, вместе с типом подсказки из указанного файла, созданного посредством работы системы 100, могут также извлекаться значения токенов для символов и значения обученных векторных представлений, которые, в процессе векторизации и токенизации, подставляются на место символов, добавленных в результате форматирования.

Кроме того, модуль 203 в еще одном частном варианте осуществления также может определять возможные целевые классы и сопоставлять им текстовые аналоги. Как указывалось выше, целевые классы (поле "ответ") не всегда могут быть изложены в текстовом виде. С учетом того, что указанный модуль 203 использует целевые классы, определенные при дообучении языковой модели на обучающем наборе данных, могут возникнуть ситуации, когда целевыми классами обучения будут являться классы, изложенные не в текстовом виде. Указанный этап работы модуля 203 может быть реализован, например, посредством перебора словаря языковой модели и/или поиском по таблице, содержащей индексные значения и текстовую интерпретацию указанных значений, хранящихся в файле словаря языковой модели. Стоит отметить, что аналогичные средства применяются в модуле 104. Так, в одном частном варианте осуществления текстовые аналоги целевых классов могут также содержаться в файле данных, содержащем языковую модель и сгенерированную подсказку. Общий принцип работы заключается в том, что каждому целевому классу (в случае, если целевые классы в файле данных содержаться не в текстовом виде) ставится в соответствии три сущности: слово, соответствующее понятию класса в естественной речи (например, "да" для положительного класса и "нет" для отрицательного), индекс данного слова в словаре входных эмбеддингов (векторных представлений), порядковый индекс класса.

Таким образом, модуль 203 осуществляет предобработку входного набора данных.

Модуль обработки 204 реализован на базе языковой модели, полученной от модуля 202, и выполнен с возможностью осуществления обработки отформатированного входного набора данных.

Общий принцип работы указанного модуля 204 заключается в обработке путем подачи на вход языковой модели отформатированного и векторизованного набора данных. Результатом обработки указанного набора (выход языковой модели) будет вектор с распределением вероятностей токенов, так называемый вектор логитов. Более подробно общий принцип работы языковой модели раскрыт в уровне технике (см., например, ссылка в Интернет: https://arxiv.org/pdf/2005.14165.pdf).

Модуль постобработки 205 может быть реализован на базе по меньшей мере одного вычислительного устройства, оснащенного соответствующим программным обеспечением, и предназначен для выбора целевых классов, полученных в результате обработки отформатированного входного набора данных дообученной языковой моделью. Так, как упоминалось выше, на выходе языковой модели содержится вектор с распределением вероятностей следующего токена, который представляет собой набор логитов для каждого токена входной последовательности. Из данного вектора модуль выбора целевых компонент отбирает только те классы, которые соответствуют токенам вербализации целевого класса (т.е. токенам из поля "ответ" обучающего набора данных). Более формально, в данном модуле 205 происходит выделение логитов, используемых для (а) вычисления вероятностей классов и (б) вычисления функции потерь. Принимает на вход вектор выходных логитов дообученной языковой модели для нескольких последовательностей, где для каждой последовательности вектор выходных логитов соответствует последнему токену. В каждом векторе логитов, поскольку его длина равна размеру словаря, выбираются компоненты, соответствующие токенам текстовых аналогов классов. Таким образом, для каждой входной последовательности выходом является единственный вектор, длина которого равна количеству классов.

Для специалиста в данной области техники очевидно, что, хотя и описанные выше модули представлены как отдельные устройства, указанные модули также могут быть объединены в составе одной системы, например системы 500.

На фиг. 3 представлена блок-схема способа 300 автоматического дообучения языковой модели для

классификации текста, который раскрыт поэтапно более подробно ниже. Указанный способ 300 заключается в выполнении этапов, направленных на обработку различных цифровых данных. Обработка, как правило, выполняется с помощью системы, например системы 100, которая также может представлять, например, сервер, компьютер, мобильное устройство, вычислительное устройство и т.д.

Как упоминалось выше, для дообучения языковой модели классификации текста обучающий набор необходимо представить в определенном виде (дополнить подсказкой), чтобы обеспечить возможность точно решать языковой моделью указанную задачу. В настоящий момент, в известном уровне техники, такие подсказки генерируются вручную, что является трудоемким процессом и не обеспечивает высокую точность результата решаемой задачи ввиду человеческого фактора. Способ 300, в частности, направлен на решение указанной проблемы.

На этапе 301 система, такая как система 100, принимает на вход языковую модель, например, от модуля 102 и обучающий входной набор данных, содержащий по меньшей мере список полей входного набора данных. Указанный этап 301 в одном частном варианте осуществления может быть выполнен модулем 101.

Как упоминалось выше, входной обучающий набор данных содержит список полей, определяющий требуемую задачу классификации. Языковая модель может представлять собой адаптируемую предобученную языковую модель, например BERT, GPT3, ELMo, Transformer и т.д. В предпочтительном варианте реализации была использована языковая модель RuGPT3-Large. Входной обучающий набор данных может быть сформирован на основе решаемой задачи классификации, например, для систем фильтрации ненормативной лексики, диалоговых ассистентов и т.д.

Поскольку изначально языковая модель не способна на классификацию текста, а способна только выдавать вероятности предсказания следующего слова для входного текста, то для решения задач классификации необходимо выполнить дообучение языковой модели.

Указанное дообучение обеспечивает повышение точности генерации ответа модели за счет правильной адаптации весов генерируемой подсказки (значений обучаемых векторных представлений символов) для конкретного типа решаемой задачи классификации. Для этого на вход языковой модели подают обучающие наборы, которые представлены таким образом, чтобы выход языковой модели являлся ответом на входной текст, т.е. формируют подсказку и дополняют обучающие наборы указанной подсказкой, путем продолжения которой языковая модель на выходе представляет ответ. Кроме того, во входном обучающем наборе содержится правильный ответ, что обеспечивает адаптацию подсказки для языковой модели в соответствии с обучающими данными и решаемой задачи классификации. Соответственно список полей обучающего набора данных содержит по меньшей мере один входной текст в определенной форме и ответ. Так, например, при решении логической задачи ответа на поиск причинно-следственной связи между текстами обучающий набор данных будет содержать следующие поля: "текст 1:" "Вася ловит рыбу", "текст 2:" "Вася дома?", "ответ:" нет,

Соответственно, одной из особенностей указанного способа 300 является возможность автоматической генерации такой подсказки для любой задачи классификации. Для этого способ 300 переходит к этапу 302.

На этапе 302 выполняют выбор формата подсказки, на основе списка полей обучающего входного набора данных, для заданной задачи классификации и выполняют форматирование обучающего входного набора данных, на основе выбранного формата подсказки, в ходе которого выполняют дополнение упомянутого набора данных символами, при этом каждый символ соответствует абстрактному псевдо-слову. Указанный этап 302 может выполняться, например, модулем 103.

На этапе 302 обучающий входной набор данных преобразовывается в строку, которая состоит из списка полей указанного набора, и дополняется символами, соответствующими псевдо-словам. Позиция и количество указанных символов определяются на основе количества входных полей набора данных. Кроме того, в одном частном варианте осуществления для типовых задач классификации формат подсказки может быть заранее сохранен в памяти системы, например системы 100, в виде шаблона. Таким образом, при форматировании обучающего набора сначала обучающий набор преобразовывается в строку и далее дополняется символами, соответствующими абстрактным псевдо-словам. Так, символ '{P}' является абстрактными псевдо-словами, а фрагмент {имя_поля} является полем обучающего набора, содержимое которого необходимо включить на данную позицию при формировании подсказки к конкретному примеру.

Так, например, в одном частном варианте осуществления при форматировании обучающего набора для решения задачи классификации текста в бинарной вопросно-ответной системе будет следующий формат: "{P}{P}{P} {текст 1}{P}{P}{P} {текст 2}{P}{P}{P}", где символ {P} соответствует абстрактному псевдо-слову, т.е. для данного символа не существует человеко-читаемого текстового аналога. Таким образом, на указанном этапе 302 формируется формат подсказки. В одном частном варианте осуществления после этапа 302 может следовать опциональный этап 303. Указанный этап 303 необходим в случае, когда токенизатор не оптимизирован для токенизации символов, соответствующих абстрактным псевдо-словам. Стоит отметить, что оптимизация токенизатора может быть осуществлена, например, до начала способа 300. При необходимости опционального этапа 303 на указанном этапе 303 добавляют в

токенизатор по меньшей мере один токен, соответствующий символу, добавленному на этапе 302, и сохраняют его индекс в словаре. Таким образом, указанный этап 303 необходим только, если в исходном токенизаторе отсутствует такое слово, как добавленный символ, что приведет к некорректной токенизации текста. Далее способ 300 также может переходить к опциональному этапу 304, указанный этап 304 может выполняться модулем 104. Как упоминалось выше, указанный этап 304 необходим в случае несоответствия формата целевых классов в обучающем входном наборе (поле "ответ"). Так, на этапе 304 сопоставляют возможным в заданной задаче классификации целевым классам, содержащимся в обучающем входном наборе данных, полученном на этапе 301, их текстовые аналоги.

Так, для бинарной вопросно-ответной системы целевые классы будут преобразованы в слова следующим образом: {положительный => "да", отрицательный => "нет"}. Для вычисления индекса данных слов в словаре входных эмбеддингов языковой модели далее осуществляют последовательно токенизацию каждого из слов ("да" и "нет"). Указанные индексы далее сохраняются в системе 100 и применяются в процессе обучения, который более подробно описан ниже.

На этапе 305 выполняют токенизацию и векторизацию отформатированного на этапе 302 обучающего входного набора данных, причем векторизация выполняется только для токенов, которые не относятся к символам.

Входной текст может быть разделен на токены. Под токеном в данном решении следует понимать последовательность символов в тексте, которая имеет значение для анализа. Стоит отметить, что определенный метод токенизации и векторизации зависит от языковой модели, используемой в заявленном решении и хранящейся в модуле 102. Например, при использовании модели RuGPT3 токенизация осуществляется методом BPE (Byte Pair Encoding), а последующая векторизация - путем замены каждого токена на его индекс в словаре языковой модели, составленном на этапе изначального обучения модели. В еще одном частном варианте осуществления токенизация может представлять собой разбиение текста на слова по пробелу между словами. Далее составляется словарь токенов фиксированного размера (например, 30000 токенов), где каждому токену сопоставляется его индекс в словаре. Кроме того, в еще одном частном варианте осуществления может выполняться векторизация каждого токена, полученного в процессе токенизации, например, с помощью прямого кодирования (one hot encoding). Так, например, при токенизации на основе алгоритма BPE каждый токен, полученный в ходе указанного процесса токенизации, представлен в словаре своим индексом, отображающим позицию в указанном словаре. Таким образом, каждый токен представляет бинарный вектор (значения 0 или 1), а единица ставится тому элементу, который соответствует номеру токена в словаре, что позволяет представить каждый токен в виде вектора фиксированной длины, соответствующей размерности словаря (например, размерности 3000 на 1). Для специалиста в данной области техники будет очевидно, что для векторизации токенов можно применять и другие алгоритмы векторизации, например алгоритмы Word2vec и т.д., не ограничиваясь.

Соответственно для добавленных символов не существует индекса в словаре, в связи с чем на данном этапе 305 их векторизация не проводится.

На этапе 306 инициализируют обучаемые векторные представления для токенизированных символов, соответствующих абстрактным псевдо-словам, и заменяют упомянутые токенизированные символы на инициализированные обучаемые векторные представления символов.

На этапе 306 для токенизированных символов задают их начальное векторное представление в словаре. Так, в одном частном варианте осуществления указанное векторное представление (обучаемые эмбеддинги) может быть задано случайным числом из словаря. Указанные заданные значения подставляются на место токенизированных символов, т.е. происходит векторизация. В ходе обучения эмбеддинги будут изменены таким образом, чтобы модель с наибольшей вероятностью выдавала верный ответ в заданной задаче.

На этапе 307 подают на вход языковой модели отформатированные обучающие входные данные в виде эмбеддингов и получают вероятности текстовых аналогов целевых классов.

На этапе 307 векторные представления отформатированного обучающего набора данных, полученные на этапе 305, объединяются с инициализированными обучаемыми векторными представлениями символов, полученных на этапе 306. Указанное объединение может быть выполнено, например, модулем инъекции 107. Объединение может быть выполнено путем вставки (инъекции) обучаемых эмбеддингов на места, соответствующие символу (символ {P}). Полученный объединенный тензор эмбеддингов затем используется в качестве входного слоя для дообучаемой языковой модели (например, для RuGPT3-Large).

Стоит отметить, что при обработке тензора эмбеддингов языковой моделью веса языковой модели фиксируются в неизменном состоянии. При обучении, несмотря на то, что входные векторные представления, в том числе и обучаемые, проходят через модель и используются при вычислении функции потерь, веса самой модели не обновляются при шаге градиентного спуска. Особенностью указанного способа 300 является возможность дообучения языковой модели (автоматического подбора подсказки) без изменения ее весов, что существенно снижает вычислительные ресурсы, необходимые на обучение и, как следствие, снижает объем памяти, требуемый для хранения адаптированной под решение задачи классификации модели.

На этапе 308 обновляют обучаемые векторные представления на основе истинных значений целевых классов для каждого обучающего объекта, итеративно повторяя этапы 307-308 до выполнения критерия останова.

Для дообучения языковой модели может быть использован модуль 108. В результате обучения будут получены оптимальные для данной задачи классификации значения обучаемых эмбеддингов, которые в дальнейшем можно будет использовать для решения задачи классификации.

На указанном этапе 308 осуществляется подбор оптимальных значений для обучаемых эмбеддингов, например, с помощью стохастического градиентного спуска. Так, например, список входных эмбеддингов, полученных на этапе 307, поступает в слои дообучаемой (адаптируемой) модели, на выходе которой содержится тензор данных с распределением вероятностей следующего токена. Из указанного тензора выбираются только вероятности, соответствующие целевым классам, и вычисляется функция потерь между выбранными вероятностями и индексом целевого класса. После чего с помощью процедуры обратного распространения ошибки вычисляются градиенты для обучаемых эмбеддингов и выполняется их обновление. В ходе обучения эмбеддинги будут меняться таким образом, чтобы математическое ожидание значения функции потерь на любом объекте из распределения, соответствующего распределению объектов обучающего набора данных, уменьшалось.

Рассмотрим в качестве примера следующий обучающий набор данных:

обучающий набор данных: [{мама мыла раму?, да}, {есть ли на марсе жизнь?, нет}].

Обучение эмбеддингов, демонстрирующее как повышается вероятность правильного класса:

GPT(<P=[0.33, 0.11, 0.1]>Мама мыла раму?) => [да: 0.1, нет: 0.5, возможно: 0.0, небо: 0.1] => вероятность правильного ответа: 0.1,

GPT(<P=[0.32, 0.12, 0.1]>Мама мыла раму?) => [да: 0.2, нет: 0.4, возможно: 0.0, небо: 0.1] => вероятность правильного ответа: 0.2,

GPT(<P=[0.35, 0.09, 0.1]>Мама мыла раму?) => [да: 0.4, нет: 0.2, возможно: 0.0, небо: 0.1] => вероятность правильного ответа: 0.4,

GPT(<P=[0.2, 0.3, 0.1]>Мама мыла раму?) => [да: 0.5, нет: 0.1, возможно: 0.0, небо: 0.1] => вероятность правильного ответа: 0.5,

GPT(<P=[0.1, 0.5, 0.1]>Мама мыла раму?) => [да: 0.5, нет: 0.1, возможно: 0.0, небо: 0.1] => вероятность правильного ответа: 0.5.

Как видно из примера, обучение эмбеддингов остановлено в связи с выполнением критерия останова. Указанный критерий останова может представлять, например, фиксированное число итераций, достижение значения заданной вероятности для целевого класса и т.д. Хотя и указанный пример обучения эмбеддингов приведен на одном обучающем примере, важно отметить, что при каждом обновлении значений эмбеддингов учитываются сразу несколько примеров из обучающего набора данных.

На этапе 309 формируют файл данных и сохраняют обученные векторные представления символов, соответствующих абстрактным псевдо-словам.

Обученные эмбедденги, полученные на этапе 309, сохраняются в файл данных.

Указанные эмбеддинги являются оптимальными для данной задачи классификации и предназначены для повышения точности языковой модели при решении последующих задач классификации схожего типа. В одном частном варианте осуществления обученные эмбеддинги (значения векторных представлений) сохраняются вместе с форматом подсказки для указанной задачи классификации. Специалисту в данной области техники очевидно, что для разных задач классификации может храниться несколько файлов с обученными эмбеддингами.

Таким образом, за счет автоматической генерации подсказки для дообучения языковой модели на решение заданной задачи классификации текста обеспечивается возможность дообучения языковой модели на малых вычислительных мощностях с сохранением высокой точности для решаемой задачи. Кроме того, указанный процесс дообучения снижает необходимый объем памяти, требуемый для хранения дообученной языковой модели (требуется хранить только обученные эмбеддинги).

На фиг. 4 представлена блок-схема способа 400 классификации текста языковой моделью, который раскрыт поэтапно более подробно ниже. Указанный способ 400 заключается в выполнении этапов, направленных на обработку различных цифровых данных. Обработка, как правило, выполняется с помощью системы, например системы 200, которая также может представлять, например, сервер, компьютер, мобильное устройство, вычислительное устройство и т.д.

На этапе 401 получают входной набор данных, соответствующий требуемой задаче классификации, в формате, на основе которого дообучалась языковая модель, и дообученную языковую модель.

Входной набор данных, соответствующий требуемой задаче классификации, в формате, на основе которого дообучалась языковая модель, представляет собой набор данных, который структурно совпадает с обучающим набором данных (содержит аналогичные поля), однако в нем отсутствует результат решаемой задачи классификации. Так, указанный входной набор данных может быть получен, например, от диалоговой системы посредством канала связи, например сети Интернет, и может содержать, например, для решаемой задачи классификации текста для ответа на бинарный вопрос набор, состоящий из двух полей: "текст:", "вопрос:". Т.е. входной набор представляет собой непосредственный текст, который

до этого не был подан в языковую модель, в формате, схожем по структуре с обучающим входным набором данных. Дообученная языковая модель может быть получена из модуля 202. Указанная дообученная языковая модель является языковой моделью, полученной в результате работы системы 100. Так, в одном частном варианте осуществления дообученная языковая модель представляет собой языковую модель со сгенерированной подсказкой. Так, например, дообученная языковая модель со сгенерированным форматом подсказки (обученными эмбеддингами) может быть получена от системы 100 после выполнения способа 300. На этапе 402 выполняют форматирование входного набора данных, дополняя его символами, причем каждый символ соответствует абстрактному псевдо-слову. Указанный этап 402 может выполняться, например, модулем 203.

На этапе 402 осуществляется преобразование входного набора данных в строку, составляя её из списка полей, который содержится в указанном наборе данных, и добавление символов между полями входного набора, причем количество и расположение символов в процессе форматирования соответствуют формату подсказки, Как было указано ранее, для разных задач классификации могут быть сгенерированы разные подсказки под соответствующий тип решаемой задачи. Так, в одном частном варианте осуществления форматирование входного набора данных осуществляется на основе шаблона для форматирования. Стоит отметить, что тип решаемой задачи, а следовательно, и тип шаблона форматирования могут поступать, например, совместно с входным набором данных. Кроме того, в еще одном частном варианте осуществления форматы шаблонов для различных задач могут храниться в памяти системы 200, например, в виде файла и могут выбираться в соответствии с типом входных полей, определяемых, например, с помощью регулярных выражений.

Общий принцип форматирования заключается в преобразовании входного набора данных в строку, составляя её из списка полей, который содержится в указанном наборе данных, и добавлении символов между полями входного набора, причем количество и расположение символов в процессе форматирования соответствуют формату подсказки. Таким образом, при форматировании входного набора сначала входной набор преобразовывается в строку и далее дополняется символами, причем количество и позиция указанных символов соответствуют количеству и позициям символов, добавленных в процессе форматирования обучающего набора модулем 103 для конкретной задачи классификации. Так, например, в одном частном варианте осуществления результатом форматирования входного набора будет следующий формат: "{P}{P}{P}{текст 1}{P}{P}{P}{текст 2}{P}{P}{P}". Причем указанный формат соответствует формату, выбранному в процессе работы системы 100 для задачи классификации. На этапе 403 выполняют токенизацию и векторизацию входного набора данных, причем символы, соответствующие абстрактным псевдо-словам, заменяются на обученные векторные представления символов, сохраненных в файле данных.

На этапе 403 осуществляются токенизация и векторизация входного набора данных, например, посредством модуля 203. Причем в ходе процесса токенизации и векторизации указанная токенизация и векторизация выполняются для всего входного набора данных, где символы, соответствующие абстрактным псевдо-словам, заменяются на обученные векторные представления символов, полученных от системы 100, а векторизация токенизированных символов осуществляется на основе значений, обученных эмбеддингов. Как указывалось выше, значения обученных эмбеддингов, полученные в результате выполнения способа 100, могут быть отправлены в систему 200 посредством канала передачи данных. Так, вместе с типом подсказки может быть отправлен файл, из которого извлекаются значения токенов для символов и значения обученных векторных представлений, которые, в процессе векторизации и токенизации, подставляются на место символов, добавленных в результате форматирования.

На этапе 404 выполняют обработку данных, полученных на этапе 403, дообученной языковой моделью, в ходе которой получают вектор логитов, отражающий вероятностное распределение классов, соответствующих словам словаря языковой модели. Так, на указанном этапе 404 отформатированный набор входных данных поступает на вход дообученной языковой модели, где происходит обработка указанных данных. На выходе языковой модели содержится тензор с распределением вероятностей следующего токена, который представляет собой набор логитов для каждого токена входной последовательности.

Рассмотрим указанный этап на примере.

Пример решения задачи определения истинности входного текста.

В качестве целевых классов для решаемой задачи из обучающего набора данных были определены следующие классы: I - текст истинный, II -текст ложный, соответственно им были присвоены следующие текстовые аналоги: I - да, II - нет. Результатом обработки дообученной языковой моделью входного набора данных, представленного в следующем виде: "текст:" "вопрос:" будет следующий вектор логитов: v = [0.1, 0, 0, 0.3, 0.5] = [в, на, да, нет, наверное]. Длина указанного вектора равняется длине словаря языковой модели, например 50000 слов.

Таким образом, на указанном этапе 404 выполняется получение вектора логитов, отражающего вероятностное распределение классов, соответствующих словам словаря языковой модели.

В качестве еще одного примера рассмотрим решение задачи сравнения количества слов "муха" и "стекло" в тексте. Целевые классы: I - слово муха встречается чаще, II - слово стекло встречается чаще. Текстовые аналоги: I - муха, II - стекло. Выход дообученной языковой модели v = [0.1, 0, 0, 0.3, 0.5] = [в,

на, муха, стекло, наверное].

Длина выходного вектора |v| = const = |V|, т.е. равна длине словаря.

На этапе 405 выбирают целевые компоненты логитов, соответствующие токенам целевых классов решаемой задачи классификации.

На указанном этапе 405 из вектора логитов, например, с помощью модуля 208 осуществляют выбор целевых компонент, которые соответствуют целевым классам. Так, продолжая пример реализации для решаемой задачи определения истинности входного текста, будут отобраны следующие целевые компоненты: [I, II] = [да, нет] = [0, 0.3].

Соответственно для другого примера сравнения количества слов "муха" и "стекло" в тексте будут отобраны следующие целевые компоненты: [I, II] = [муха, стекло] = [0, 0.3].

На этапе 406 определяют из целевых компонентов логитов, полученных на этапе е), компоненту логита, отражающую наибольшее значение вероятности принадлежности к целевому классу.

На указанном этапе 406 из отобранных целевых компонент выбирают компоненту с наибольшей вероятностью. Так, продолжая пример задачи определения истинности входного текста, второй целевой класс имеет значение вероятности, большее, чем первый, следовательно, II > I = нет > да => ответ II (текст ложный). Для примера сравнения количества слов "муха" и "стекло" в тексте соответственно II > I = стекло > муха => ответ II (слово стекло встречается чаще).

На этапе 407 генерируют ответ в текстовой форме, соответствующий выбранной на этапе 406 компоненте.

Так, на указанном этапе 407 текстовый аналог выбранного целевого класса, имеющего наибольшую вероятность, может быть отправлен в систему, для которой решалась поставленная задача классификации. Так, указанными системами могут являться диалоговые ассистенты, системы фильтрации ненормативной лексики, вопросно-ответные системы и т.д. Передача сгенерированного ответа может осуществляться по каналу связи, такому как Интернет. Кроме того, в одном частном варианте осуществления сгенерированный ответ может быть сохранен в памяти системы, такой как система 500. Таким образом, в вышеприведенных материалах были описаны системы и способы автоматического дообучения языковой модели для решения задач классификации и классификация текста языковой моделью.

Теперь рассмотрим примеры реализации заявленного технического решения. Как упоминалось выше, заявленная группа изобретений выполнена с возможностью решения прикладных задач классификации. Адаптация изобретения под конкретную задачу классификации выполняется изменением формата подсказки и соответственно изменением процесса форматирования исходного текста и целевых классов, представляющих ответ на решаемую задачу. Рассмотрим различные варианты реализации заявленного решения на наборе задач из RussianSuperGLUE (доступно по ссылке, найдено в Интернет: https://russiansuperglue.com/tasks/).

Для примера возьмем задание Terra (задача определения, учитывая два текстовых фрагмента, выводится ли (может быть выведено) значение одного текста из другого текста). Для решения указанной задачи языковая модель дообучалась на обучающем наборе данных, содержащем 2016 примеров. При обучении модели RuGPT3 X1 (доступна по ссылке: https://huggingface.co/sberbank-ai/rugpt3large_based_on_gpt2). Удалось добиться значительного сокращения затраченных ресурсов на обучение, так как выполнялось обучение только 3.2 миллионов параметров (подбор обучаемых эмбеддингов) вместо 760 миллионов параметров (подбор весов языковой модели для решения указанной задачи). При этом качество (точность) языковой модели превысило качество языковой модели, обученной стандартными методами (76.1 accuracy вместо 65.4 accuracy). Пример задания: "текст:" "Автор поста написал в комментарии, что прорвалась канализация", "вывод:" "Автор поста написал про канализацию. "ответ:" "Связь есть".

На фиг. 5 представлен пример общего вида вычислительной системы 500, которая обеспечивает реализацию заявленных способов 300 и 400 или является частью компьютерной системы, например системы 100 или системы 200, и/или реализует модули указанных систем 100 и 200. Кроме того, указанная система 500 может являться сервером, персональным компьютером, частью вычислительного кластера, обрабатывающим необходимые данные для осуществления заявленного технического решения. В общем случае система 500 содержит такие компоненты, как один или более процессоров 501, по меньшей мере одну память 502, средство хранения данных 503, интерфейсы ввода/вывода 504, средство В/В 505, средство сетевого взаимодействия 506, которые объединяются посредством универсальной шины.

Процессор 501 выполняет основные вычислительные операции, необходимые для обработки данных при выполнении способа 300 и способа 400. Процессор 501 исполняет необходимые машиночитаемые команды, содержащиеся в оперативной памяти 302.

Память 502, как правило, выполнена в виде ОЗУ и содержит необходимую программную логику, обеспечивающую требуемый функционал.

Средство хранения данных 503 может выполняться в виде HDD, SSD дисков, рейд массива, флэш-памяти, оптических накопителей информации (CD, DVD, MD, Blue-Ray дисков) и т.п. Средства 503 позволяют выполнять долгосрочное хранение различного вида информации, например языковых моделей, обученных эмбеддингов и т.п. Для организации работы компонентов системы 500 и организации работы

внешних подключаемых устройств применяются различные виды интерфейсов В/В 504. Выбор соответствующих интерфейсов зависит от конкретного исполнения вычислительного устройства, которые могут представлять собой, не ограничиваясь, PCI, AGP, PS/2. IrDa, FireWire, LPT, COM, SATA, IDE, Lightning. USB (2.0, 3.0, 3.1, micro, mini, type C). TRS/Audio jack (2.5. 3.5. 6.35). HDMI, DVI, VGA, Display Port, RJ45, RS232 и т.п. Выбор интерфейсов 504 зависит от конкретного исполнения системы 500. которая может быть реализована на базе широкого класса устройств, например персональный компьютер, мейнфрейм, ноутбук, серверный кластер, тонкий клиент, смартфон, сервер и т.п.

В качестве средств В/В данных 505 может использоваться клавиатура, джойстик, дисплей (сенсорный дисплей), монитор, сенсорный дисплей, тачпад, манипулятор мышь, световое перо, стилус, сенсорная панель, трекбол, динамики, микрофон, средства дополненной реальности, оптические сенсоры, планшет, световые индикаторы, проектор, камера, средства биометрической идентификации (сканер сетчатки глаза, сканер отпечатков пальцев, модуль распознавания голоса) и т.п. Средства сетевого взаимодействия 506 выбираются из устройств, обеспечивающих сетевой прием и передачу данных, например Ethernet карту, WLAN/Wi-Fi модуль, Bluetooth модуль, BLE модуль, NFC модуль, IrDa, RFID модуль, GSM модем и т.п. С помощью средств 505 обеспечивается организация обмена данными между, например, системой 500, представленной в виде сервера, и вычислительным устройством пользователя, на котором могут отображаться полученные данные (ответ в вопросно-ответной системе) по проводному или беспроводному каналу передачи данных, например WAN, PAN, ЛВС (LAN), Интранет, Интернет, WLAN, WMAN или GSM. Конкретный выбор элементов системы 500 для реализации различных программно-аппаратных архитектурных решений может варьироваться с сохранением обеспечиваемого требуемого функционала.

Представленные материалы раскрывают предпочтительные примеры реализации технического решения и не должны трактоваться как ограничивающие иные, частные примеры его воплощения, не выходящие за пределы испрашиваемой правовой охраны, которые являются очевидными для специалистов соответствующей области техники. Таким образом, объем настоящего технического решения ограничен только объемом прилагаемой формулы.

ФОРМУЛА ИЗОБРЕТЕНИЯ

1. Способ автоматического дообучения языковой модели для классификации текста, выполняющийся по меньшей мере одним вычислительным устройством и содержащий этапы, на которых:

a) принимают языковую модель и обучающий входной набор данных, содержащий по меньшей мере список полей входного набора данных;

b) выбирают формат подсказки, на основе списка полей обучающего входного набора данных, для заданной задачи классификации и выполняют форматирование обучающего входного набора данных, на основе выбранного формата подсказки, в ходе которого выполняют дополнение упомянутого набора данных символами, при этом каждый символ соответствует абстрактному псевдо-слову;

c) выполняют токенизацию и векторизацию отформатированного на этапе b) обучающего входного набора данных, причем векторизация выполняется только для токенов, которые не относятся к символам;

d) инициализируют обучаемые векторные представления для токенизированных символов, соответствующих абстрактным псевдо-словам, и заменяют упомянутые токенизированные символы на инициализированные обучаемые векторные представления символов;

e) подают на вход языковой модели отформатированный обучающий входной набор данных в виде векторных представлений и получают вероятности текстовых аналогов целевых классов;

f) обновляют обучаемые векторные представления, на основе истинных значений целевых классов для каждого обучающего объекта, итеративно повторяя этапы e)-f) до выполнения критерия останова;

g) формируют файл данных и сохраняют обученные векторные представления символов, соответствующих абстрактным псевдо-словам.

2. Способ по п.1, характеризующийся тем, что форматирование обучающего входного набора данных выполняется с использованием шаблона для форматирования.

3. Способ по п.2, характеризующийся тем, что шаблон для форматирования выбирается на основе решаемой задачи классификации.

4. Способ по п.3, характеризующийся тем, что задача классификации представляет собой по меньшей мере:

i) бинарные вопросно-ответные системы;

ii) бинарная классификация на распознавание причинно-следственной связи между двумя предложениями;

iii) бинарная классификация на выбор одной из двух альтернатив;

iv) задача на машинное чтение в форме бинарной классификации;

v) задача бинарной классификации на распознавание причинно-следственных связей между посылкой и гипотезой.

5. Способ классификации текста языковой моделью, дообученной согласно этапам способа по любому из пп.1-4, выполняющийся по меньшей мере одним вычислительным устройством и содержащий этапы, на которых:

a) получают входной набор данных, соответствующий требуемой задаче классификации, в формате, на основе которого дообучалась языковая модель;

b) выполняют форматирование входного набора данных, дополняя его символами, причем каждый символ соответствует абстрактному псевдо-слову;

c) выполняют токенизацию и векторизацию входного набора данных, причем символы, соответствующие абстрактным псевдо-словам, заменяются на обученные векторные представления символов, сохраненных в файле данных;

d) выполняют обработку данных, полученных на этапе c), дообученной языковой моделью, в ходе которой получают вектор логитов, отражающий вероятностное распределение классов, соответствующих словам словаря языковой модели;

e) выбирают целевые компоненты логитов, соответствующие токенам целевых классов решаемой задачи классификации;

f) определяют из целевых компонентов логитов, полученных на этапе e), компоненту логита, отражающую наибольшее значение вероятности принадлежности к целевому классу;

g) генерируют ответ в текстовой форме, соответствующий выбранной на этапе f) компоненте.

6. Система автоматического дообучения языковой модели для классификации текста, содержащая:

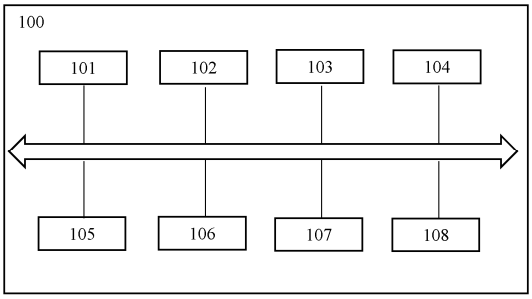
по меньшей мере один процессор;

по меньшей мере одну память, соединенную с процессором, которая содержит машиночитаемые инструкции, которые при их выполнении по меньшей мере одним процессором обеспечивают выполнение способа по любому из пп.1-4.

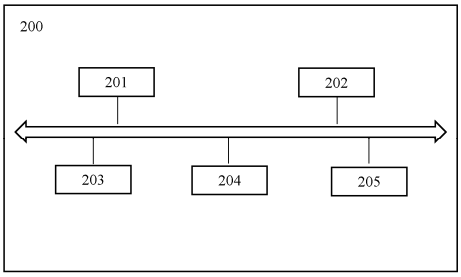
7. Система классификации текста дообученной языковой моделью, содержащая:

по меньшей мере один процессор;

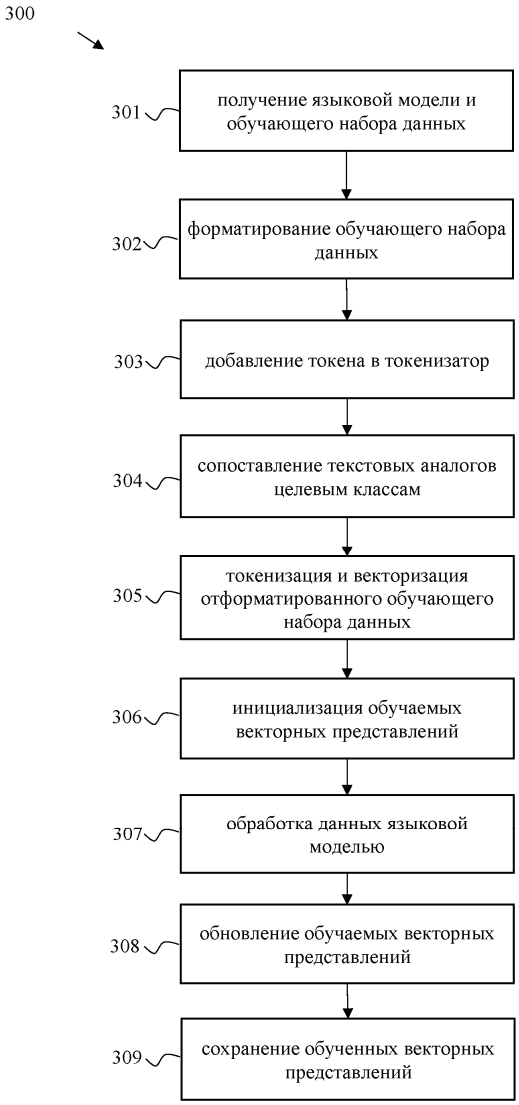
по меньшей мере одну память, соединенную с процессором, которая содержит машиночитаемые инструкции, которые при их выполнении по меньшей мере одним процессором обеспечивают выполнение способа по п.5.



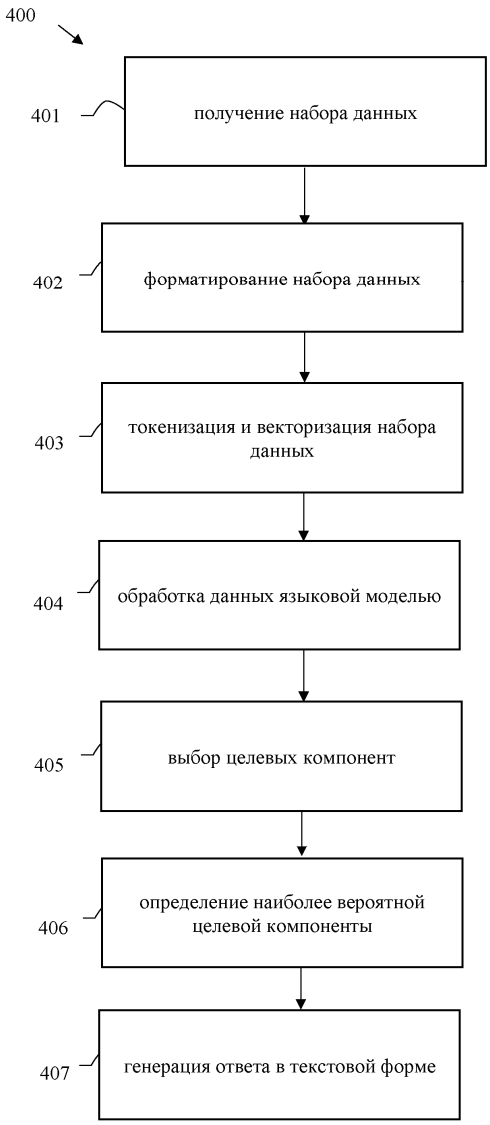
Фиг. 1



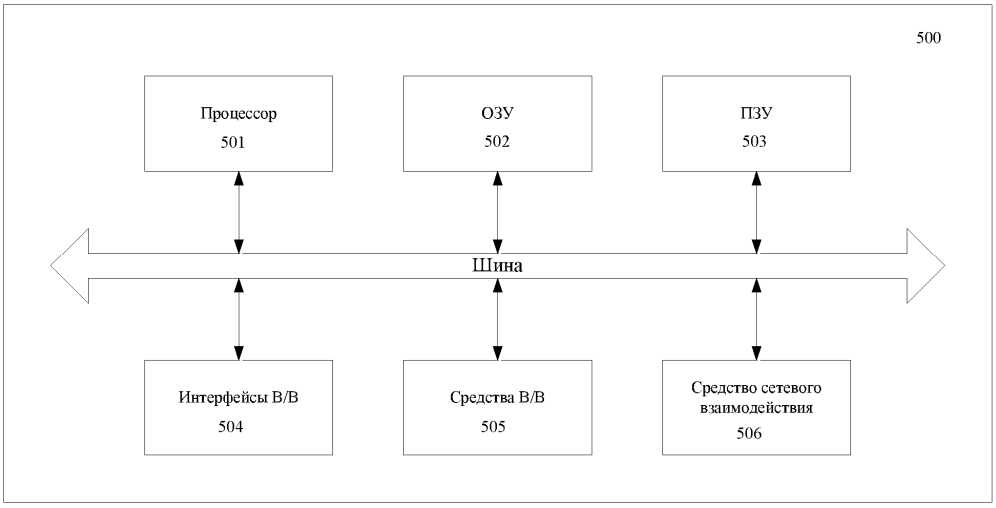
Фиг. 2


300
301
получение языковой модели и обучающего набора данных
302
форматирование обучающего набора данных
303
добавление токена в токенизатор
304
сопоставление текстовых аналогов целевым классам
305
токенизация и векторизация отформатированного обучающего набора данных
306
инициализация обучаемых векторных представлений
307
обработка данных языковой моделью
308
обновление обучаемых векторных представлений
309
сохранение обученных векторных представлений

Фиг. 3


400
401
получение набора данных
402
форматирование набора данных
403
токенизация и векторизация набора данных
404
обработка данных языковой моделью
405
выбор целевых компонент
406
определение наиболее вероятной целевой компоненты
407
генерация ответа в текстовой форме

Фиг. 4


500
Процессор
501
ОЗУ
502
ПЗУ
503
Шина
Интерфейсы В/В
504
Средства В/В
505
Средство сетевого взаимодействия
506

Фиг. 5