УДК 004.4+004.6+004.8+004.9

ПРИМЕНЕНИЕ МАШИННОГО ОБУЧЕНИЯ К ЗАДАЧЕ ГЕНЕРАЦИИ ПОИСКОВЫХ ЗАПРОСОВ

А. М. Гусенков[1, [0000-0003-4019-7322]], А. Р. Ситтикова[2, [0000-0002-9539-764X]]

[1,2]*Казанский* (Приволжский) федеральный университет, Казань, Россия

[1]gusenkov.a.m@gmail.com, [2]sitti.alina@mail.ru

Аннотация

Исследованы две модификации рекуррентных нейронных сетей: сети с долгой краткосрочной памятью и сети с управляемым рекуррентным блоком с добавлением механизма внимания к обеим сетям, а также модель Transformer в задаче генерации запросов к поисковым системам. В качестве модели Transformer использована модель GPT-2 от OpenAI, которая обучалась на запросах пользователей. Проведен латентно-семантический анализ для определения семантических сходств между корпусом пользовательских запросов и запросов, генерируемых нейронными сетями. Для проведения анализа корпус был переведен в формат bag of words, к нему применена модель TFIDF, проведено сингулярное разложение. Семантическое сходство вычислялось на основе косинусной меры. Также для более полной оценки применимости моделей к задаче был проведен экспертный анализ для оценки связности слов в искусственно созданных запросах.

Ключевые слова: *обработка естественного языка, генерация естественного языка, машинное обучение, нейронные сети.*

ВВЕДЕНИЕ

Генерация естественного языка – это процесс создания осмысленных фраз и предложений в форме естественного языка. Среди основных применяемых подходов можно выделить два алгоритма создания текстов: методы на основе правил и методы на основе машинного обучения. Первый подход позволяет добиться высокого качества текстов, но требует знания правил языка и времени для разработки [1], в то время как второй подход зависит только от данных для обучения, но часто допускает грамматические и семантические ошибки в создаваемых текстах [2].

В настоящее время активно исследуется метод генерации текстов с помощью нейронных сетей; один из самых популярных алгоритмов – рекуррентные нейронные сети [3]. Вторая ведущая архитектура – модель Transformer [3]. Эти архитектуры рассматривались в решении задачи генерации поисковых запросов.

Цель данной статьи – изучить вышеупомянутые архитектуры, проанализировать их качество и применимость к этой задаче. Использование автоматически генерируемых запросов для поисковых систем актуально, поскольку большинство компаний не выдает поисковые запросы бесплатно, а поисковая система в процессе разработки должна быть протестирована. Также полученные запросы можно использовать для повышения эффективности и оптимизации поисковой системы.

Были использованы поисковые запросы от пользователей AOL (America Online), которые были анонимно размещены в интернете в 2006 г. Хотя названная компания не идентифицировала своих пользователей, личная информация присутствовала во многих запросах [4], чего компании сейчас пытаются избежать. Были предложены алгоритмы, помогающие сохранить анонимность пользователей, но существует вопрос о том, имеют ли данные, которые можно безопасно публиковать, практическую пользу. Для решения этой проблемы предлагается использовать автоматически генерируемые запросы.

ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

Алгоритмы генерации текста на естественном языке активно изучаются и используются во многих программных системах, поэтому на данный момент ведется большое количество исследований в этой области.

Один из первых применяемых подходов – это система шаблонов для заполнения пробелов. Она используется в текстах, которые имеют предопределенную структуру, и, если необходимо заполнить небольшой объем данных, этот подход может автоматически заполнять пробелы данными, полученными из электронных таблиц, баз данных и т. д. Примером такого подхода является Microsoft Word mailmerge [5].

Вторым шагом было добавление к первому подходу языков программирования общего назначения, которые поддерживают сложные условные выражения, циклы и т. д. Этот подход более эффективный и полезный, но отсутствие языковых возможностей затрудняет создание систем, которые могут генерировать качественные тексты.

Следующим шагом в развитии систем на основе шаблонов является добавление грамматических функций на уровне слов, связанных с морфологией и правописанием. Такие функции значительно упрощают создание грамматически правильных текстов. Далее системы динамически создают предложения из представлений значений, которые они должны передать. Это означает, что системы могут обрабатывать необычные случаи без необходимости явного написания кода для каждого случая и значительно лучше генерируют высококачественного тексты на «микроуровне». Наконец, на следующем этапе развития системы могут генерировать хорошо структурированные документы, актуальные для пользователей. Например, текст, который должен быть убедительным, может быть основан на моделях аргументации и изменения поведения [5].

После перехода от шаблонов к генерации динамического текста потребовалось много времени, чтобы добиться удовлетворительных результатов. Если рассматривать создание текстов на естественном языке как подраздел обработки естественного языка, то существует ряд наиболее развитых алгоритмов: цепи Маркова [6], рекуррентные нейронные сети, сети с долгой краткосрочной памятью и модель Transformer. Существуют инструменты для генерации текста, основанные на этих методах, например, коммерческие Arria NLG PLC, AX Semantics, Yseop и другие, а также программы с открытым исходным кодом Simplenlg, GPT, GPT-2, BERT, XLNet.

Кроме того, в настоящее время исследуется использование генеративно-состязательных сетей для генерации текста, поскольку они показывают отличные результаты в задаче генерации изображений [7].

СБОР ДАННЫХ

В качестве обучающих данных для нейронных сетей были выбраны пользовательские запросы на английском языке из поисковой системы AOL 2006 года. Исследователи стараются избегать использования этих данных в своих работах,

так как они могут считаться разоблачающими, но в данной статье используются только тексты самих запросов, без идентификаторов пользователей и сайтов, на которые они перешли, то есть без использования личной информации. Исходные данные представлены в виде, показанном на рис. 1.

```
Query   QueryTime       ItemRank        ClickURL
carbol tunnel   2006-03-01 01:01:21
how to install a glue down floor        2006-03-01 07:13:45     2       http://doityourself.com
how to install a glue down floor        2006-03-01 07:13:45     8       http://www.homerenovationguide.com
how to install a glue down floor        2006-03-01 07:13:45     9       http://www.hardwoodinstaller.com
how to install a glue down floor        2006-03-01 07:35:01     20      http://www.ehow.com
how to install a glue down floor        2006-03-01 07:43:50     26      http://www.hoskinghardwood.com
mapquest        2006-03-01 19:40:11     1       http://www.mapquest.com
indian projectile points        2006-03-02 21:12:10
indian projectile points        2006-03-02 21:13:02
indian projectile points        2006-03-02 21:13:03     1       http://www.utexas.edu
indian projectile points        2006-03-02 21:13:03     6       http://www.iath.virginia.edu
indian projectile points        2006-03-02 21:22:40
indian projectile points        2006-03-02 21:22:42
indian projectile points        2006-03-02 21:22:46     16      http://www.mnsu.edu
indian projectile points        2006-03-02 21:22:46     18      http://www.madison.k12.wi.us
```

Рис. 1. Исходные данные для обучения.

Запросы длиной более 32 слов и ошибочные запросы, не содержащие информации, были удалены из корпуса. Повторяющиеся запросы и запросы, содержащие названия веб-сайтов, также были удалены, поскольку они не являются примерами естественного языка. Всего были случайным образом выбраны 100 тыс. запросов для обучения. На рис. 2 показаны примеры данных после предварительной обработки.

```
i can love you like that
breyer traditional horse models
waffle irons
home made structures
dominion a prequel to the exorcist
what is a liability
capstone turbine
danville ill
statetax
old club men ties
piciculture
pagan jewelry
unicoi county memorial hospital
```

Рис. 2. Данные после предварительной обработки.

Запросы были разделены на символы-токены, каждому символу было присвоено натуральное число, весь корпус был закодирован с помощью этого словаря.

РЕКУРРЕНТНЫЕ СЕТИ

Рекуррентные нейронные сети (Recurrent Neural Networks, RNN) – это семейство нейронных сетей, в которых связи между элементами образуют направленную последовательность [8]. Они могут использовать свою внутреннюю память для обработки последовательностей произвольной длины, а также хорошо распознают зависимости между токенами. Однако рекуррентные сети учатся медленно, и их способность запоминать длинные зависимости ограничена из-за проблемы затухания градиентов [9].

Были реализованы два типа рекуррентных сетей, которые чаще всего используются в задаче генерации текстов на естественном языке – сети с долгой краткосрочной памятью (Long Short-Term Memory Network) [10] и сети с управляемым рекуррентным блоком (Gated Recurrent Unit) [11]. Исследования показали, что эти типы сетей имеют сопоставимую точность, а в зависимости от решаемой задачи одна сеть может быть точнее другой.

СЕТИ С ДОЛГОЙ КРАТКОСРОЧНОЙ ПАМЯТЬЮ

Сеть с долгой краткосрочной памятью (Long Short-Term Memory Networks, LSTM) – это система глубокого обучения, которая позволяет избежать проблемы затухания и взрывного роста градиентов [10]. Сети LSTM могут запоминать значительно более длинные последовательности символов. Они используют вентили, которые являются внутренними механизмами, которые могут контролировать информационный поток. На рис. 3 представлен общий вид ячейки LSTM.

В каждой ячейке сети есть 3 вентиля: входной, выходной и вентиль забывания. Вектор вентиля забывания вычисляется по формуле

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f),$$

где x_t – входной вектор, h_{t-1} – выходной вектор предыдущей ячейки, σ – сигмоидальная функция, W_f, U_f, b_f – матрицы весов и вектор смещения.

Далее входной вентиль обновляет состояние ячейки по следующим формулам:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \qquad \hat{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c).$$

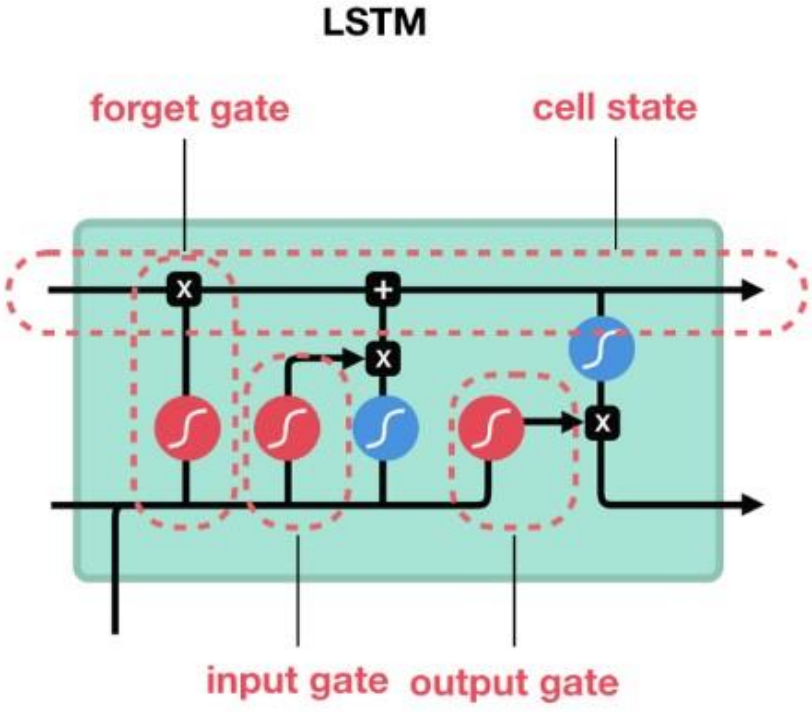

Рис. 3. Ячейка LSTM.

Затем вычисляется новое значение состояния ячейки:

$$c_t = f_t \circ c_{t-1} + i_t \circ \hat{c}_t,$$

где c_{t-1} – состояние предыдущей ячейки. Наконец, выходной вентиль решает, каким должно быть следующее скрытое состояние по формулам:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o),\ h_t = o_t \circ \tanh(c_t).$$

Результаты передаются в следующую ячейку.

СЕТИ С УПРАВЛЯЕМЫМ РЕКУРРЕНТНЫМ БЛОКОМ

Вторая реализованная модель – это сеть с управляемым рекуррентным блоком (Gated Recurrent Unit, GRU) – новое поколение рекуррентных нейронных сетей, похожих на сети с долгой краткосрочной памятью [11]. Однако по сравнению с LSTM этот тип сетей имеет меньше параметров, и поэтому эти модели обучаются быстрее. У GRU всего два вентиля: обновления и сброса. На рис. 4 представлен общий вид ячейки сети GRU.

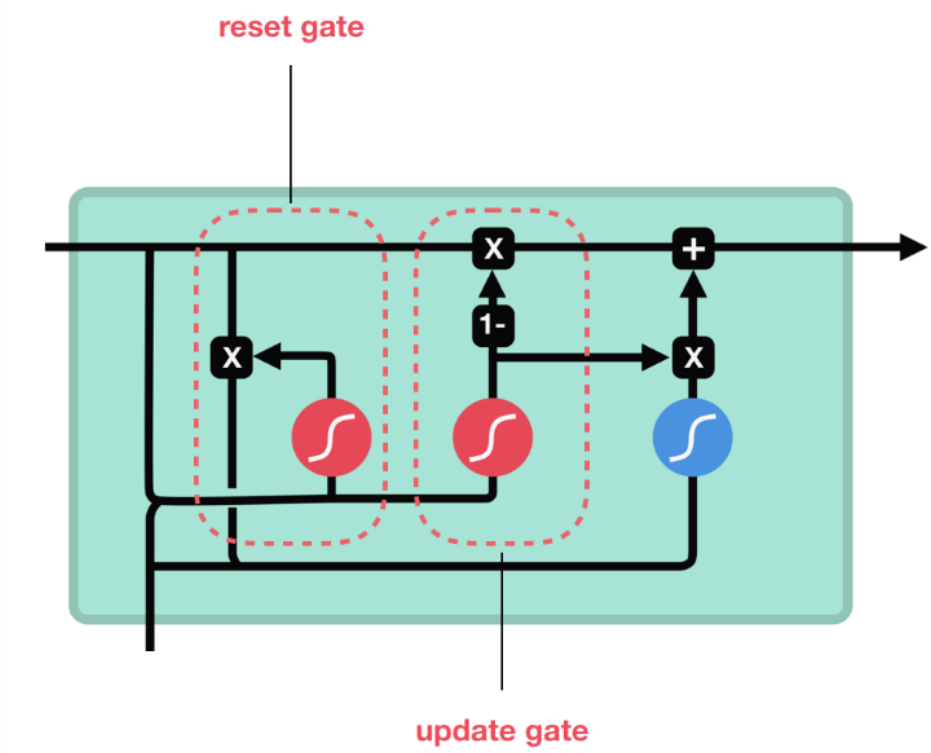

Рис. 4. Ячейка GRU

Вентиль обновления действует подобно входному вентилю и вентилю забывания в LSTM и вычисляется по формуле

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z).$$

Вентиль сброса рассчитывается по формуле

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r).$$

Выходной вектор ячейки GRU определяется следующим образом:

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \sigma(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h).$$

МЕХАНИЗМ ВНИМАНИЯ В РЕКУРРЕНТНЫХ НЕЙРОННЫХ СЕТЯХ

Механизм внимания – это метод, используемый в нейронных сетях для выявления зависимостей между частями входных и выходных данных [12]. Механизм внимания позволяет модели определять важность каждого слова для задачи прогнозирования путем их взвешивания при создании представления текста. Был использован следующий подход с одним параметром на входной канал [13]:

$$e_t = h_t w_a,\ a_t = \frac{\exp(e_t)}{\sum_{i=1}^{T} \exp(e_i)},\ v = \sum_{i=1}^{T} a_i h_i\ .$$

Здесь h_t – представление слова в момент времени t, w_a – матрица весов для слоя внимания, a_t – оценки внимания для каждого момента времени, а v – вектор представления текста.

РЕАЛИЗАЦИЯ РЕКУРРЕНТНЫХ СЕТЕЙ

Все нейронные сети были реализованы на Python 3.7 в пакете Google Collab [14], поскольку он позволяет использовать графические процессоры, что значительно сокращает время обучения моделей. Для реализации нейронных сетей мы выбрали библиотеку Keras [15], которая представляет собой высокоуровневую надстройку над TensorFlow. Эта библиотека значительно упрощает разработку нейронных сетей, так как в ней уже есть готовые реализации основных слоев, функции активации и потери. Был использован оптимизатор Adam (Adaptive Moment Estimation [16]) – алгоритм, в котором скорость обучения регулируется для каждого параметра. Также использована функция Learning Rate Scheduler [17] в качестве callback, которая позволяет вычислять коэффициент скорости обучения с помощью определенной функции. Примерная архитектура модели представлена на рис. 5.

Предварительно обработанные данные были разделены на обучающую и валидационную выборки, которые составили 80% и 20% корпуса соответственно. Обучающие данные передаются на вход слою Embedding, который преобразует числа в векторы, отражающие соответствия между последовательностями символов и проекции этих последовательностей. Полученные представления передаются на вход первому слою LSTM (GRU), его выходные данные передаются второму слою LSTM (GRU), и тем же образом – третьему. Затем выходные данные из слоя Embedding и этих трех слоев объединяются и передаются слою Attention-WeightedAverage. Вектор представления, полученный из слоя внимания, представляет собой высокоуровневое кодирование всего текста, которое используется в качестве входных данных для конечного полносвязного слоя с активацией Softmax для классификации [13].

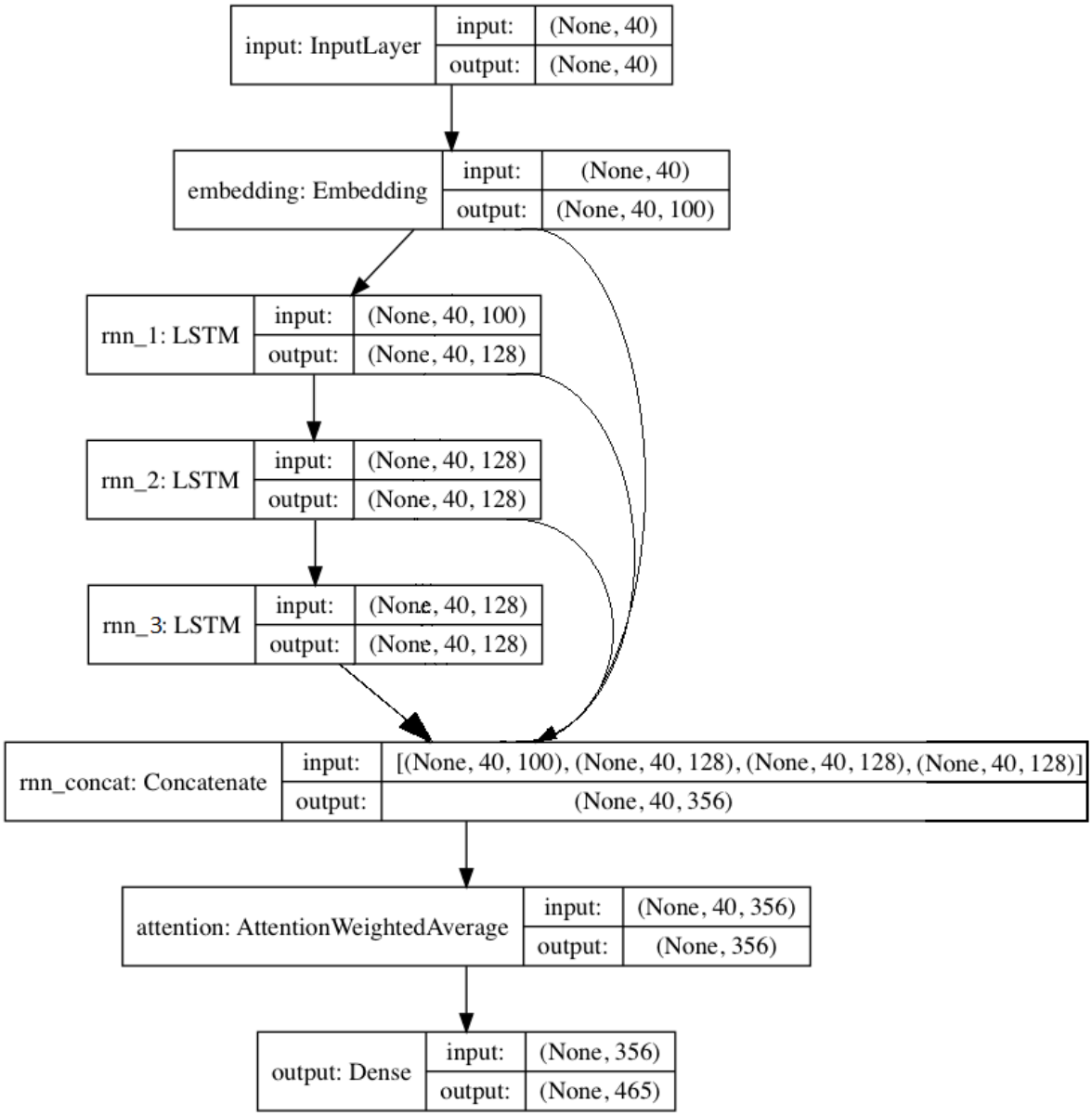

Рис. 5. Архитектура модели.

Для проверки, насколько хорошо или плохо модель обучилась за конкретную эпоху, вычисляется функция потерь Categorical Cross-Entropy по формуле

$$L(y, \hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}\left(y_{ij}\log\left(\hat{y}_{ij}\right)\right)$$

где $\hat{y}$ – предсказанные значения.

Были проведены эксперименты с изменением количества слоев LSTM (GRU) в модель (2 и 3 слоя), а также с добавлением слоя Dropout после слоя Embedding,

который случайным образом исключает заданное количество нейронов, чтобы предотвратить переобучение сети и лучше обобщить модель. Сети с 3 рекуррентными слоями и Dropout показали лучший результат.

Также были обучены двунаправленные модели этих сетей. Двунаправленная рекуррентная нейронная сеть – это модель, предложенная в 1997 году Майком Шустером и Кулдип Паливал [18], которая позволяет рассматривать контекст слова не только слева от него, но и справа от последовательности. Общий вид двунаправленных нейронных сетей представлен на рис. 6.

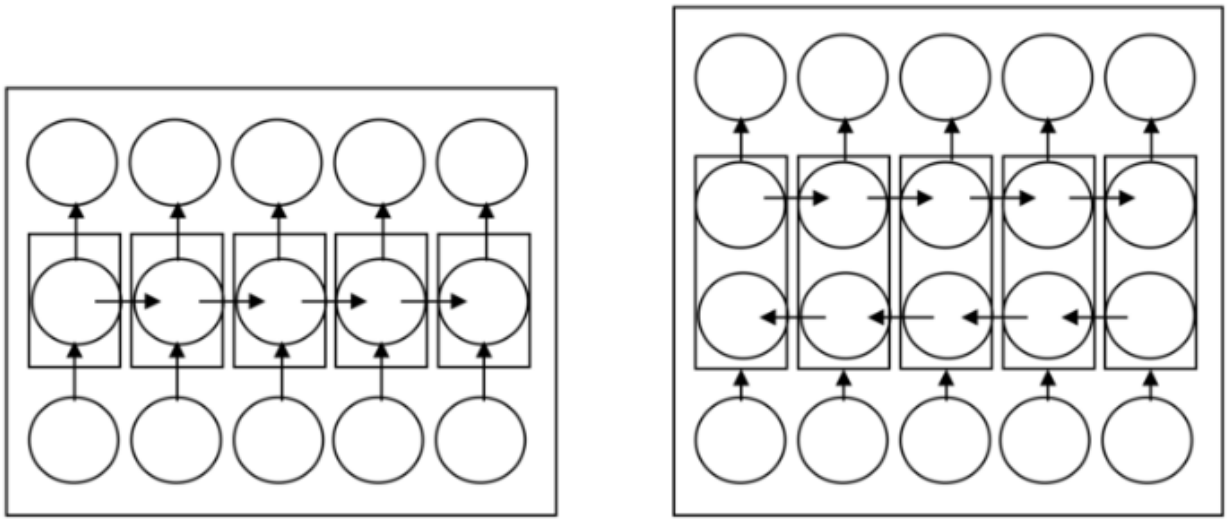

Рис. 6. Общий вид: а) Однонаправленная нейронная сеть;
б) Двунаправленная нейронная сеть

В случае задачи генерации поисковых запросов двунаправленная модель показала себя лучше однонаправленной; полученные значения функции потерь после обучения модели в течение 30 эпох представлены в таблице 1.

Таблица 1. Значения функции потерь

	LSTM	GRU	Bi-LSTM	Bi-GRU
Loss	1,48	1,58	1,30	1,37
Validation Loss	1,6	1,62	1,56	1,57

Значение функции потерь уменьшилось на обучающих данных, однако на валидационных данных улучшение было менее значительным, что позволяет предположить, что двунаправленная модель в этом задаче не так хорошо обучается, а скорее «запоминает» последовательности символов.

С помощью реализованной модели были сгенерированы запросы с разной «температурой». Это параметр, который влияет на шанс выбора маловероятного символа.

ТРАНСФОРМЕР

Transformer – это модель глубокого обучения, которая была представлена в 2017 году [19]. Общий вид его архитектуры показан на рис. 7.

Трансформеры состоят из стэков равного количества энкодеров и декодеров. Энкодеры обрабатывают входные последовательности и кодируют данные для отражения информации о них и их признаках. Декодеры делают обратное, они обрабатывают полученную от энкодера информацию и генерируют выходные последовательности. Все энкодеры имеют одинаковую структуру и состоят из двух слоёв: внутреннее внимание (Self-Attention) и нейронная сеть с прямой связью (feed-forward neural network). Входная последовательность, поступающая в энкодер, сначала проходит через слой внутреннего внимания, помогающий энкодеру посмотреть на другие слова во входном предложении при кодировании конкретного слова. Выходные данные этого слоя отправляются в нейронную сеть с прямой связью. Такая же сеть независимо применяется к каждому слову. Декодер также содержит два этих слоя, но между ними есть дополнительный слой внимания, который позволяет декодеру определить релевантные части входного предложения.

Внутреннее внимание позволяет модели видеть зависимости между обрабатываемым словом и другими словами во входной последовательности, которые помогают лучше закодировать слово.

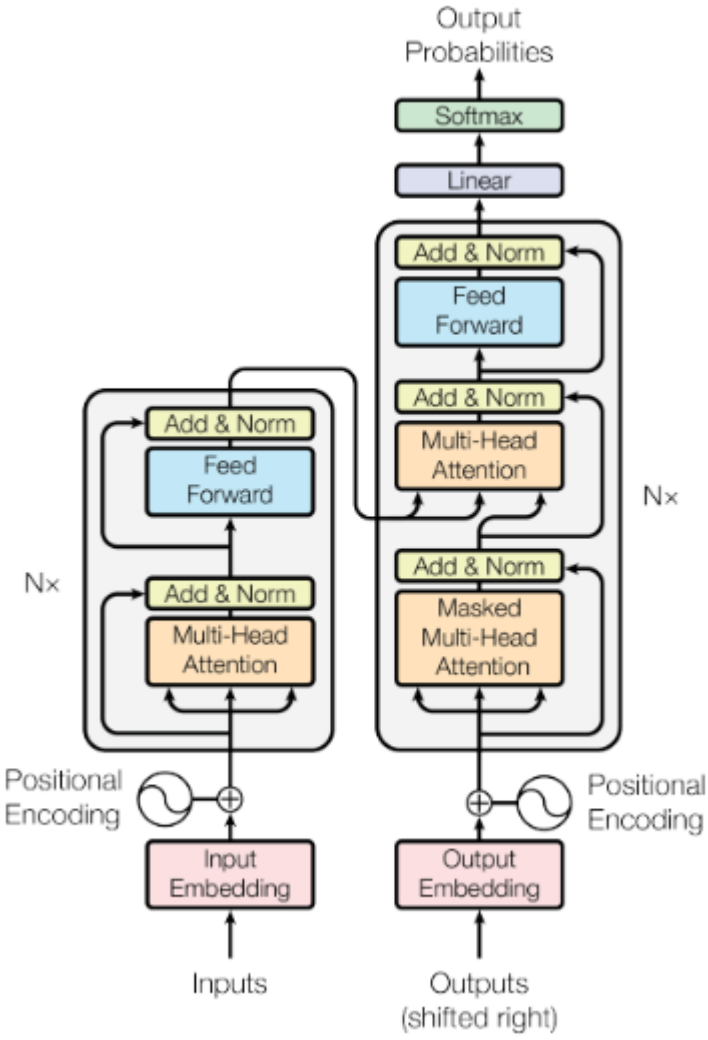

Рис. 7. Архитектура трансформера

После всех декодеров используется полносвязный слой Softmax, который преобразует полученные значения в вероятности, из которых затем выбирается наибольшее значение, а соответствующее ему слово становится выходом для этого временного шага.

АРХИТЕКТУРА GPT-2

GPT-2 – это большая языковая модель на основе модели Transformer, созданная некоммерческой компанией OpenAI, с количеством параметров от 117 миллионов до 1,5 миллиарда, обученная на наборе данных из 8 миллионов веб-страниц [20]. GPT-2 обучается с простой целью: предсказать следующее слово, учитывая все предыдущие слова в некотором тексте.

GPT-2 построена с использованием только блоков декодеров, которые имеют ту же структуру, что и в описанной выше общей модели Transformer.

В качестве входных данных GPT-2 использует не слова, а токены, полученные с помощью метода Byte Pair Encoding (BPE). Это метод сжатия данных, в котором наиболее распространенные пары последовательных байтов слов заменяются байтами, которых нет в этих словах [23]. Этот метод обеспечивает баланс между представлениями на уровне символов и слов, что позволяет ему обрабатывать большие корпуса данных.

Внутреннее внимание в GPT-2 также использует маскирование, которое блокирует информацию от токенов справа от вычисляемой позиции.

РЕАЛИЗАЦИЯ GPT-2

Была использована модель GPT-2 среднего размера с 345 млн параметров, состоящая из 24 блоков декодеров.

Модель была дообучена с помощью fine-tuning на корпусе поисковых запросов на английском языке, которые были также использованы для обучения рекуррентных нейронных сетей. С помощью полученной модели были сгенерированы поисковые запросы.

Была использована реализация модели, доступная по ссылке https://github.com/nshepperd/gpt-2. Модель была обучена на 1000 шагов.

ЛАТЕНТНО-СЕМАНТИЧЕСКИЙ АНАЛИЗ

Латентно-семантический анализ (Latent Semantic Analysis, LSA) – это метод обработки естественного языка для анализа зависимостей между коллекциями документов и терминами, содержащимися в них [24].

Метод использует терм-документную матрицу, которая описывает частоту появления терминов в коллекции документов. Элементы такой матрицы могут быть взвешены, например, с помощью TF-IDF: вес каждого элемента матрицы пропорционален количеству раз, когда термин встречается в каждом документе, и обратно пропорционален количеству раз, когда термин встречается во всех документах коллекции. После составления терм-документной матрицы проводится её сингулярное разложение, т. е. она представляется в виде $A = USV^T$, где мат-

рицы *U* и *V* – ортогональные, а *S* – диагональная матрица, значения которой называются сингулярными значениями матрицы *A*. Такое разложение отражает основную структуру зависимостей, присутствующих в исходной матрице, позволяя игнорировать шумы [25].

РЕАЛИЗАЦИЯ ЛАТЕНТНО-СЕМАНТИЧЕСКОГО АНАЛИЗА

Для проведения латентно-семантического анализа была использована библиотека gensim для Python [26]. Был создан корпус из 10000 документов, содержащий «эталонные» поисковые запросы, написанные людьми. Из него затем были удалены часто встречающиеся служебные слова английского языка (предлоги, артикли) и слова, встречающийся один раз, так как они не помогут вычислить семантическую связь между документами. С помощью класса Dictionary библиотеки gensim был создан словарь со словами и их индексами, затем с помощью метода doc2bow этого класса все документы представляются в формате «мешок слов» (bag of words). Модель TFIDF применена к полученному корпусу данных, а класс LsiModel проводит сингулярное разложение. Запросы, сгенерированные с помощью нейронных сетей, были токенизированы и с помощью словаря, созданного на «эталонном» корпусе, трансформированы в формат «мешок слов». Наконец, с помощью класса MatrixSimilarity вычисляются семантические сходства между этими корпусами с использованием косинусной меры.

РЕЗУЛЬТАТЫ ОЦЕНКИ СГЕНЕРИРОВАННЫХ ЗАПРОСОВ

Сравнивая каждый документ, в данном случае запрос, с документами из корпуса с реальными запросами, метод возвращает значение от –1 до 1, отражающее семантическое сходство документов. Результаты анализа приведены в таблице 2.

Таблица 2. Результаты латентно-семантического анализа

	GRU	LSTM	Fine-tuned GPT-2
Среднее значение	0,0065	0,006	0,0035
Среднее кол-во значений >0,7	16	14	9
Среднее кол-во значений > 0	4000	4659	2684

Корпус реальных запросов разнообразен, поэтому среднее значение результата сравнения каждого сгенерированного документа со всеми документами из «эталонного» корпуса незначительно отличается от нуля. При этом для каждого запроса, искусственно созданного с помощью сетей GRU и LSTM, существует в среднем 16 и 14 семантически близких документов, когда значения больше 0,7, а для дообученной модели GPT-2 это количество составило 9 документов. Также для каждого запроса, сгенерированного моделью GPT-2, из 10000 сравниваемых документов 2684 имеют значение больше 0, а для сетей LSTM и GRU – 4659 и 4000 соответственно. Из этого можно сделать вывод, что LSTM и GRU при генерации запросов использовали больше слов, семантически похожих на слова из обучающих данных, чем GPT-2. Это имеет смысл, так как первые две модели были обучены с нуля на входных данных, в то время как основное обучение последней модели происходило на совершенно ином корпусе, она была только дообучена с помощью метода fine-tuning, чтобы генерировать запросы, подходящие по структуре. Также важно принимать во внимание, что сравнение проводилось с 10 тысячами «эталонных» запросов, хотя модели обучались на 100 тысячах, соответственно не все зависимости были учтены, однако полученных значений достаточно для анализа.

Результаты анализа показывают, что сгенерированные запросы имеют схожую семантику с корпусом реальных запросов пользователей, но при этом не повторяют их буквально, то есть являются новыми по смыслу запросами.

Сети GRU и LSTM были обучены посимвольно и могли сгенерировать несуществующие слова, поэтому было решено проверить их. С помощью корпуса, содержащего более 466 тысяч английских слов, доступного по ссылке https://github.com/dwyl/english-words, каждое слово из запросов было проверено на существование. В запросах, сгенерированных сетью GRU, не было найдено 141 слово из 4431, а в запросах модели LSTM – 166 из 4325. Ненайденные слова содержали опечатки или ошибки, которые модели запомнили. Следовательно, возможно, стоит предобрабатывать данные, исправляя опечатки и ошибки такого рода. Однако запросы с опечатками могут быть полезны в зависимости от задачи, в которой они будут применяться. Так, например, при их использовании для те-

стирования новой поисковой системы или её оптимизации они будут более актуальными с опечатками, так как имеют большую схожесть с реальными пользовательскими запросами.

В силу того, что нейронные сети не могут понимать смысл предложения, хоть и часто находят верные зависимости между токенами, был проведен экспертный (ручной) анализ для оценки качества сгенерированных поисковых запросов.

Из запросов, созданных каждой моделью, случайным образом были выбраны 100 запросов. Было определено, имеет ли смысл каждый поисковый запрос и похож ли он на реальный возможный запрос пользователя. Следует отметить, что такая оценка субъективна. Запросы считались «хорошими», если слова в них были согласованы друг с другом.

Результаты анализа приведены в таблице 3.

Таблица 3. Результаты экспертной оценки поисковых запросов

	GRU	LSTM	Fine-tuned GPT-2
Хороший запрос	72	73	81
Плохой запрос	28	27	19

Из таблицы видно, что сети GRU и LSTM показали практически одинаковые результаты, а GPT-2 немного лучше. В ходе анализа было замечено, что модель GPT-2 генерирует более короткие запросы, чем две другие модели.

Результаты проведенных анализов показали, что сети GRU и LSTM имеют приблизительно одинаковое качество при решении задачи генерации поисковых запросов, а модель GPT-2 оказалась хуже в автоматическом анализе, но лучше при экспертной оценке. Следовательно, эта модель подходит лучше для генерации поисковых запросов, поскольку значимость экспертной оценки выше автоматической, хотя для более точных результатов стоит провести эту оценку с помощью других экспертов.

ЗАКЛЮЧЕНИЕ

В ходе работы исследованы ведущие модели, используемые для генерации текстов на естественном языке, для решения задачи генерации запросов к поисковым системам, а также проведен их сравнительный анализ. Полностью реализованы две нейронные сети: сеть с долгой кратковременной памятью и сеть с управляемым рекуррентным блоком. Исследована архитектура GPT-2, основанная на модели Transformer, она также была дообучена с помощью корпуса реальных запросов пользователей.

Латентно-семантический анализ показал, что модель GPT-2 имеет результаты хуже, чем две другие сети. Однако автоматические метрики оценки сгенерированного текста не всегда отражают качество модели, так как на данный момент невозможно оценить осмысленность текстов с помощью алгоритма. Для решения этой проблемы также был проведен экспертный анализ сгенерированных текстов, по результатам которого модель GPT-2 оказалась лучше двух других моделей. При этом сети LSTM и GRU показали приблизительно одинаковое качество по результатам всех проведенных анализов.

СПИСОК ЛИТЕРАТУРЫ

1. *Van Deemter K., Krahmer E., Theune M.* Real vs. template-based natural language generation: a false opposition?
URL: https://wwwhome.ewi.utwente.nl/~theune/PUBS/templates-squib.pdf

2. *Xie Z.* Neural Text Generation: A Practical Guide.
URL: https://arxiv.org/pdf/1711.09534.pdf

3. A Comprehensive Guide to Natural Language Generation, 2019.
URL: https://medium.com/sciforce/a-comprehensive-guide-to-natural-language-generation-dd63a4b6e548

4. *Arrington M.* AOL proudly releases massive amounts of user search data, 2006.
URL: https://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/

5. *Reiter E.* NLG vs Templates: Levels of Sophistication in Generating Text, 2016.
URL: https://ehudreiter.com/2016/12/18/nlg-vs-templates

6. *Gagniuc P.* Markov Chains: From Theory to Implementation and Experimentation, 2017. USA, NJ: John Wiley & Sons.

7. *Press O., Bar A., Bogin B., Berant J., Wolf L.* Language Generation with Recurrent Generative Adversarial Networks without Pre-training.
URL: https://arxiv.org/pdf/1706.01399.pdf

8. *Williams R.J., Hinton G.E., Rumelhart D.E.* Learning representations by back-propagating errors. URL: http://www.cs.utoronto.ca/~hinton/absps/naturebp.pdf

9. *Hochreiter S., Bengio Y., Frasconi P., Schmidhuber J.* Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies.
URL: https://www.bioinf.jku.at/publications/older/ch7.pdf

10. *Hochreiter S., Schmidhuber J.* Long-Short Term Memory.
URL: http://web.archive.org/web/20150526132154/;
URL: http://deeplearning.cs.cmu.edu/pdfs/Hochreiter97_lstm.pdf

11. *Heck J., Salem F.* Simplified Minimal Gated Unit Variations for Recurrent Neural Networks. URL: https://arxiv.org/abs /1701.03452

12. *Bahdanau D., Cho K.m Bengio Y.* Neural Machine Translation by Jointly Learning to Align and Translate. URL: https://arxiv.org/pdf/1409.0473.pdf

13. *Felbo B., Mislove A., Søgaard A., Rahwan I., Lehmann S.* Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. URL: https://arxiv.org/pdf/1708.00524.pdf

14. *Bisong E.* Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform, 2019. Apress, Berkeley, CA.

15. *Chollet F. Keras,* 2015. URL: https://keras.io

16. *Kingma D., Ba J.* Adam: A Method for Stochastic Optimization.
URL: https://arxiv.org/abs/1412.6980

17. Learning Rate Scheduler.
URL: https://keras.io/api/callbacks/learning_rate_ scheduler/

18. *Schuster M., Paliwal K.* Bidirectional recurrent neural networks.
URL: https://www.researchgate.net/publication/3316656_Bidirectional_recurrent_neural_networks

19. *Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A., Kaizer L., Polosukhin I.* Attention Is All You Need. URL: https://arxiv.org/pdf/1706.03762.pdf

20. *Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I.* Language Models Are Unsupervised Multitask Learners.
URL: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf

21. *Devlin J., Chang M., Lee K., Toutanova K.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
URL: https://arxiv.org/pdf/1810.04805.pdf

22. *Brown T., Mann B., Ryder N., Subbiah M., Kaplan J.* Language Models Are Few-Shot Learners. URL: https://arxiv.org/abs/2005.14165

23. *Gage P.* A New Algorithm for Data Compression.
URL: https://www.derczynski.com/papers/archive/BPE_Gage.pdf

24. *Deerwester S., Harshman R.* Indexing by Latent Semantic Analysis.
URL: https://www.cs.bham.ac.uk/ ~pxt/IDA/lsa_ind.pdf

25. *Nakov P.* Getting Better Results with Latent Semantic Indexing.
URL:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.6406
&rep=rep1&type=pdf

26. *Rehurek R., Sojka P.* Software Framework for Topic Modelling with Large Corpora // Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. University of Malta. 2010.

APPLYING MACHINE LEARNING TO THE TASK OF GENERATING SEARCH QUERIES

A. M. Gusenkov[1, [0000-0003-4019-7322]], A. R. Sittikova[2, [0000-0002-9539-764X]]

[1,2]Kazan Federal University

[1]gusenkov.a.m@gmail.com , [2]sitti.alina@mail.ru

Abstract

In this paper we research two modifications of recurrent neural networks – Long Short-Term Memory networks and networks with Gated Recurrent Unit with the addition of an attention mechanism to both networks, as well as the Transformer model in the task of generating queries to search engines. GPT-2 by OpenAI was used as the Transformer, which was trained on user queries. Latent-semantic analysis was carried

out to identify semantic similarities between the corpus of user queries and queries generated by neural networks. The corpus was converted into a bag of words format, the TFIDF model was applied to it, and a singular value decomposition was performed. Semantic similarity was calculated based on the cosine measure. Also, for a more complete evaluation of the applicability of the models to the task, an expert analysis was carried out to assess the coherence of words in artificially created queries.

Keywords*: natural language processing, natural language generation, machine learning, neural networks.*

REFERENCES

1. *Van Deemter K., Krahmer E., Theune M.* Real vs. template-based natural language generation: a false opposition?
URL: https://wwwhome.ewi.utwente.nl/~theune/PUBS/templates-squib.pdf

2. *Xie Z.* Neural Text Generation: A Practical Guide.
URL: https://arxiv.org/pdf/1711.09534.pdf

3. A Comprehensive Guide to Natural Language Generation, 2019.
URL: https://medium.com/sciforce/a-comprehensive-guide-to-natural-language-generation-dd63a4b6e548

4. *Arrington M.* AOL proudly releases massive amounts of user search data, 2006.
URL: https://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/

5. *Reiter E.* NLG vs Templates: Levels of Sophistication in Generating Text, 2016.
URL: https://ehudreiter.com/2016/12/18/nlg-vs-templates

6. *Gagniuc P.* Markov Chains: From Theory to Implementation and Experimentation, 2017. USA, NJ: John Wiley & Sons.

7. *Press O., Bar A., Bogin B., Berant J., Wolf L.* Language Generation with Recurrent Generative Adversarial Networks without Pre-training.
URL: https://arxiv.org/pdf/1706.01399.pdf

8. *Williams R.J., Hinton G.E., Rumelhart D.E.* Learning representations by back-propagating errors. URL: http://www.cs.utoronto.ca/~hinton/absps/naturebp.pdf

9. *Hochreiter S., Bengio Y., Frasconi P., Schmidhuber J.* Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies.

URL: https://www.bioinf.jku.at/publications/older/ch7.pdf

10. *Hochreiter S., Schmidhuber J.* Long-Short Term Memory.
URL: http://web.archive.org/web/20150526132154/;
URL: http:// deeplearning.cs.cmu.edu/pdfs/Hochreiter97_lstm.pdf

11. *Heck J., Salem F.* Simplified Minimal Gated Unit Variations for Recurrent Neural Networks. URL: https://arxiv.org/abs /1701.03452

12. *Bahdanau D., Cho K.m Bengio Y.* Neural Machine Translation by Jointly Learning to Align and Translate. URL: https://arxiv.org/pdf/1409.0473.pdf

13. *Felbo B., Mislove A., Søgaard A., Rahwan I., Lehmann S.* Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. URL: https://arxiv.org/pdf/1708.00524.pdf

14. *Bisong E.* Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform, 2019. Apress, Berkeley, CA.

15. *Chollet F. Keras,* 2015. URL: https://keras.io

16. *Kingma D., Ba J.* Adam: A Method for Stochastic Optimization.
URL: https://arxiv.org/abs/1412.6980

17. Learning Rate Scheduler.
URL: https://keras.io/api/callbacks/learning_rate_ scheduler/

18. *Schuster M., Paliwal K.* Bidirectional recurrent neural networks.
URL: https://www.researchgate.net/publication/ 3316656_Bidirectional_recurrent_neural_networks

19. *Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A., Kaizer L., Polosukhin I.* Attention Is All You Need. URL: https://arxiv.org/pdf/1706.03762.pdf

20. *Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I.* Language Models Are Unsupervised Multitask Learners.
URL: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf

21. *Devlin J., Chang M., Lee K., Toutanova K.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
URL: https://arxiv.org/pdf/1810.04805.pdf

22. *Brown T., Mann B., Ryder N., Subbiah M., Kaplan J.* Language Models Are Few-Shot Learners. URL: https://arxiv.org/abs/2005.14165

23. *Gage P.* A New Algorithm for Data Compression.
URL: https://www.derczynski.com/papers/archive/BPE_Gage.pdf

24. *Deerwester S., Harshman R.* Indexing by Latent Semantic Analysis.
URL: https://www.cs.bham.ac.uk/ ~pxt/IDA/lsa_ind.pdf

25. *Nakov P.* Getting Better Results with Latent Semantic Indexing.
URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.6406&rep=rep1&type=pdf

26. *Rehurek R., Sojka P.* Software Framework for Topic Modelling with Large Corpora // Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. University of Malta. 2010.

СВЕДЕНИЯ ОБ АВТОРАХ

ГУСЕНКОВ Александр Михайлович – к. т. н., доцент Института вычислительной математики и информационных технологий Казанского (Приволжского) федерального университета. Области научных интересов: технологии извлечения знаний, обработка естественных языков, большие данные, интеллектуальный анализ данных.

Alexander M. GUSENKOV – assistant professor, Institute of Computational Mathematics and Information Technologies of Kazan Federal University. Ph.D. Current scientific interests: knowledge extraction technologies, Natural Language Processing, big data, data mining.

email: gusenkov.a.m@gmail.com

СИТТИКОВА Алина Рафисовна – бакалавр, Институт вычислительной математики и информационных технологий Казанского (Приволжского) федерального университета.

Alina R. SITTIKOVA – bachelor, Institute of Computational Mathematics and Information Technologies of Kazan Federal University.

sitti.alina@mail.ru

Материал поступил в редакцию 10 ноября 2020 года