Генерация поисковых запросов на основе нейронных сетей

А.М. Гусенков[1], А.Р. Ситтикова[1]

[1] *Казанский федеральный университет, Казань, Россия*

Аннотация. Исследованы модификации рекуррентных нейронных сетей – сети с долгой краткосрочной памятью (Long Short-Term Memory) и сети с управляемым рекуррентным блоком (Gated Recurrent Unit) с добавлением механизма внимания в обе сети, а также модель Transformer в задаче генерации запросов к поисковым системам. Проведены латентно-семантический анализ для выявления семантического сходства между корпусом пользовательских запросов и запросов, сгенерированных нейронными сетями, и экспертный анализ для оценки согласованности слов в искусственно созданных запросах.

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

Generation of search queries with neural networks

A.M. Gusenkov[1], A.R. Sittikova[1]

[1] *Kazan Federal University, Kazan, Russia*

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. Latent-semantic analysis was carried out to identify semantic similarities between the corpus of user queries and queries generated by neural networks. Also, 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

Введение

Генерация естественного языка – процесс создания осмысленных фраз и предложений в форме естественного языка. Среди алгоритмов создания текстов можно выделить два основных подхода: методы,

основанные на правилах, и методы, основанные на машинном обучении. Первый подход позволяет добиться высокого качества текстов, но требует знания правил языка и является трудоёмким в разработке [1], в то время как второй подход зависит только от обучающих данных, но чаще допускает грамматические и семантические ошибки в создаваемых текстах [2].

В настоящее время активно исследуется метод генерации текстов с помощью нейронных сетей, одним из наиболее популярных алгоритмов этого метода являются рекуррентные нейронные сети [3]. Вторая лидирующая архитектура – модель Transformer [3]. Именно эти архитектуры и рассмотрены для решения задачи генерации поисковых запросов.

Целью данной работы является исследование упомянутых выше архитектур, анализ их качества и применимости к данной задаче, то есть оценка насколько искусственно созданные запросы похожи на реальные запросы пользователей и возможность применения первых вместо вторых. Автоматически сгенерированные запросы могут быть использованы при тестировании поисковых систем, а также для повышения их эффективности и оптимизации.

В работе использованы поисковые запросы пользователей компании AOL (America Online), которые были анонимно выложены в свободный доступ в интернет в 2006 году и, хотя компания не идентифицировала пользователей, личная информация присутствовала во многих запросах [4], чего сейчас компании стараются избежать. Были предложены алгоритмы, которые помогают сохранить анонимность пользователей, однако стоит вопрос о том, имеют ли данные, которые можно безопасно публиковать, практическую пользу. Для решения этой проблемы предлагается использовать автоматически сгенерированные запросы.

Обзор предметной области

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

Один из первых подходов – система шаблонов fill-in-the-gap. Он 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 др. Примером такого подхода является Microsoft Word mailmerge [5].

Вторым шагом стало добавление к первому подходу языков программирования общего назначения, которые поддерживают сложные условные выражения, циклы и т.д. Такой подход является более мощным и

полезным, однако отсутствие языковых возможностей затрудняет создание систем, которые могут генерировать качественные тексты.

Следующий этап развития систем, основанных на шаблонах, – добавление грамматических функций на уровне слов, которые имеют дело с морфологией и орфографией. Такие функции значительно упрощают создание грамматически верных текстов. Далее, системы динамически создают предложения из представлений значений, которые они должны передавать. Это означает, что системы могут справляться с необычными случаями без необходимости явного написания кода для каждого случая, и значительно лучше генерируют высококачественные тексты на «микроуровне». Наконец, на следующем этапе развития системы могут генерировать хорошо структурированные документы, релевантные для пользователей. Например, текст, который должен быть убедительным, может основываться на моделях аргументации и изменения поведения [5].

После перехода от шаблонов к динамической генерации текста потребовалось много времени для достижения удовлетворительных результатов. Если рассматривать генерацию текстов на естественном языке как подраздел обработки естественного языка, то существует ряд наиболее разработанных алгоритмов – Марковские цепи [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.

Также в настоящее время исследуется использование генеративно-состязательных сетей для генерации текстов, так как они показывают отличные результаты в задаче генерации изображений [7].

Сбор данных

В качестве данных для обучения нейронной сети выбраны запросы пользователей на английском языке в поисковой системе AOL 2006 года. Исследователи опасаются использовать эти данные в своих работах, так как они могут считаться разоблачающими, однако в данной работе используются только тексты самих запросов без ID пользователей и сайтов, на которые они перешли, то есть без использования персональной информации. Исходные данные представлены в виде, показанном рис. 1.

Из корпуса были удалены запросы длиннее 32 слов и ошибочные запросы, не содержащие информации. Также были удалены дублируемые запросы и запросы, содержащие названия сайтов, так как они не являются примерами естественного языка. Всего для обучения случайным образом выбрано 100 тысяч запросов. На рис. 2 представлены примеры данных после предобработки.

```
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. Исходный вид данных для обучения

```
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 предобработки

Данные разделены на токены-символы, каждому символу сопоставлено натуральное число, весь корпус закодирован с помощью полученного словаря.

Рекуррентные сети

Рекуррентные нейронные сети (RNN) – семейство нейронных сетей, где связи между элементами образуют направленную последовательность [8]. Они могут использовать свою внутреннюю память для обработки последовательностей произвольной длины, а также хорошо выделяют зависимости между токенами.

Однако рекуррентные сети медленно обучаются, а также их способность запоминать длинные зависимости ограничена из-за проблемы затухания градиентов [9].

В работе реализованы два вида рекуррентных сетей, которые наиболее часто применяются в задаче генерации текстов на естественном языке – сеть с долгой краткосрочной памятью [10] и сеть с управляемым рекуррентным блоком [11]. Исследования показали, что эти виды сетей имеют сопоставимую точность, в зависимости от задачи одна сеть может быть точнее другой.

Сети с долгой кратковременной памятью

Сеть с долгой кратковременной памятью (англ. Long short term memory, LSTM) представляет собой систему глубинного обучения, при реализации которой удалось обойти проблему затухания или взрывного роста градиентов [10]. Сети LSTM могут запоминать значительно более длинные последовательности символов. Они используют вентили – внутренние механизмы, которые могут регулировать поток информации. На рис. 4 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.

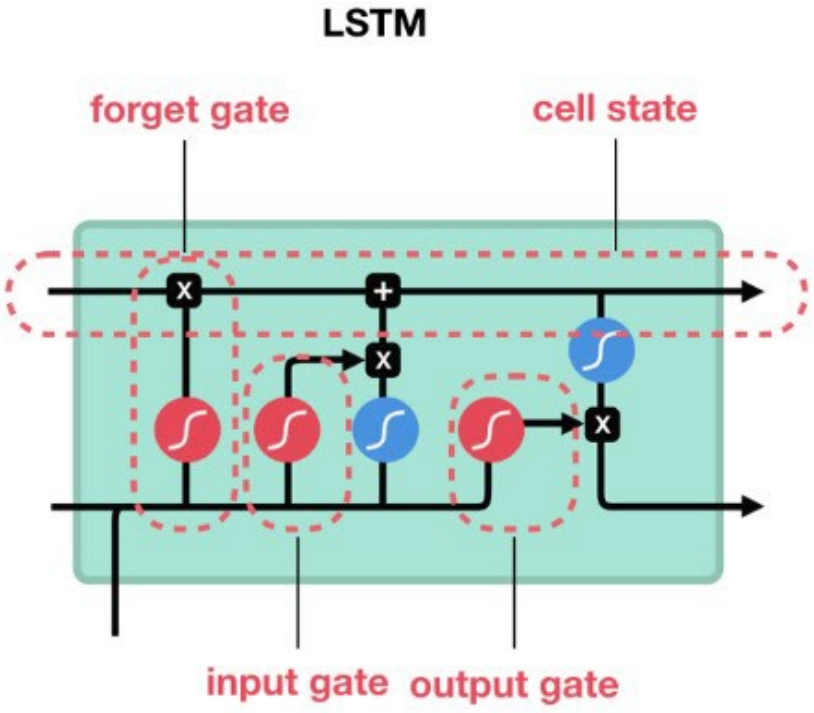

Рис. 4. Ячейка LSTM

Далее входной вентиль обновляет состояние ячейки по следующим формулам:

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

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:

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

Полученные результаты передаются следующей ячейке.

Сети с управляемым рекуррентным блоком

Вторая реализованная модель – сеть с управляемым рекуррентным блоком (англ. Gated Recurrent Units, GRU) – новое поколение рекуррентных нейронных сетей, похожа на сеть с долгой кратковременной памятью [11]. Однако по сравнению с LSTM у этого вида сетей меньше параметров, поэтому и обучаются эти модели быстрее. У GRU всего 2 вентиля: вентиль обновления и сброса. На рис. 5 представлен общий вид ячейки сети GRU.

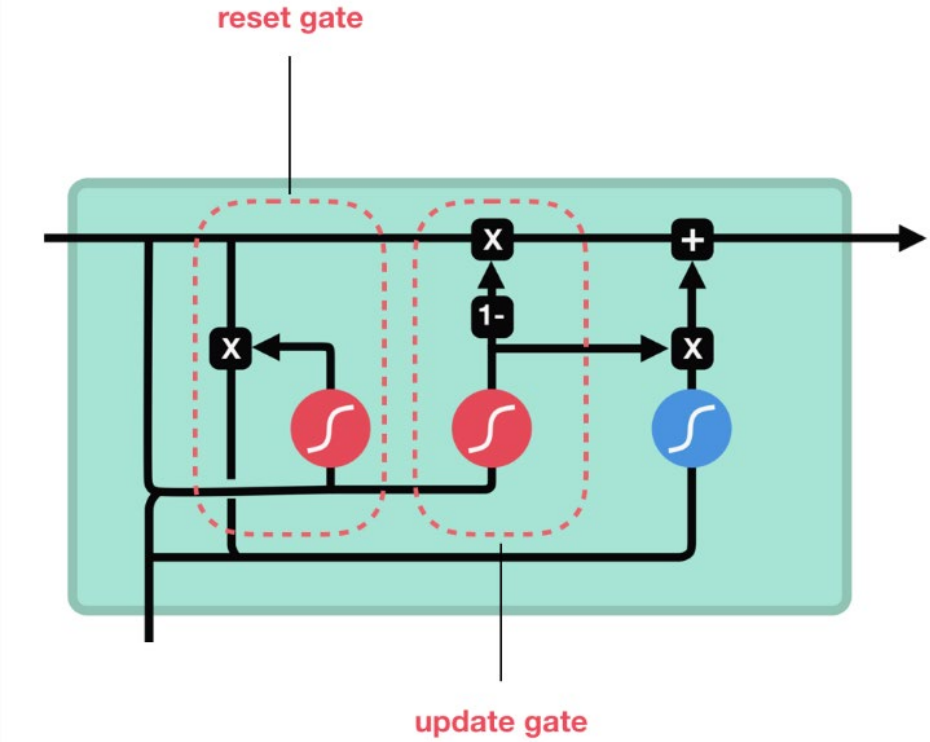

Рис. 5. Ячейка GRU

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 вычисляется 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 (англ. attention mechanism, attention model) — техника, используемая в нейронных сетях для поиска взаимосвязей между частями входных и выходных данных [12].

Механизм внимания позволяет модели определять важность каждого слова для задачи прогнозирования, взвешивая их при построении представления текста. Нами 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

Нейронные сети реализованы на языке Python 3.7 в Google Collab [14], так как в нём имеется возможность использовать графические процессоры, которые значительно ускоряют обучение моделей. Для реализации нейронных сетей выбрана библиотека Keras [15], представляющая собой высокоуровневую надстройку над TensorFlow. Эта библиотека значительно упрощает разработку нейронных сетей, так как в ней уже есть готовые реализации основных слоёв, функций активации и потерь. Использован оптимизатор Adam (Adaptive Moment Estimation) [16] — алгоритм, в котором скорость обучения настраивается 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 приведена на рис. 6.

Предобработанные данные были разделены на обучающую и валидационную выборки, которые составляли 80% и 20% корпуса соответственно. Обучающие данные подаются на вход слою Embedding, который преобразует числа в вектора, которые отражают соответствия между последовательностями символов и проекции этих последовательностей. Полученные представления подаются на вход первому слою LSTM (GRU), его выходные данные передаются второму слою LSTM (GRU), и тем же образом третьему. Далее выходные данные из слоя Embedding и этих трёх слоёв объединяются и подаются на вход слою AttentionWeightedAverage. Вектор представления, полученный из слоя

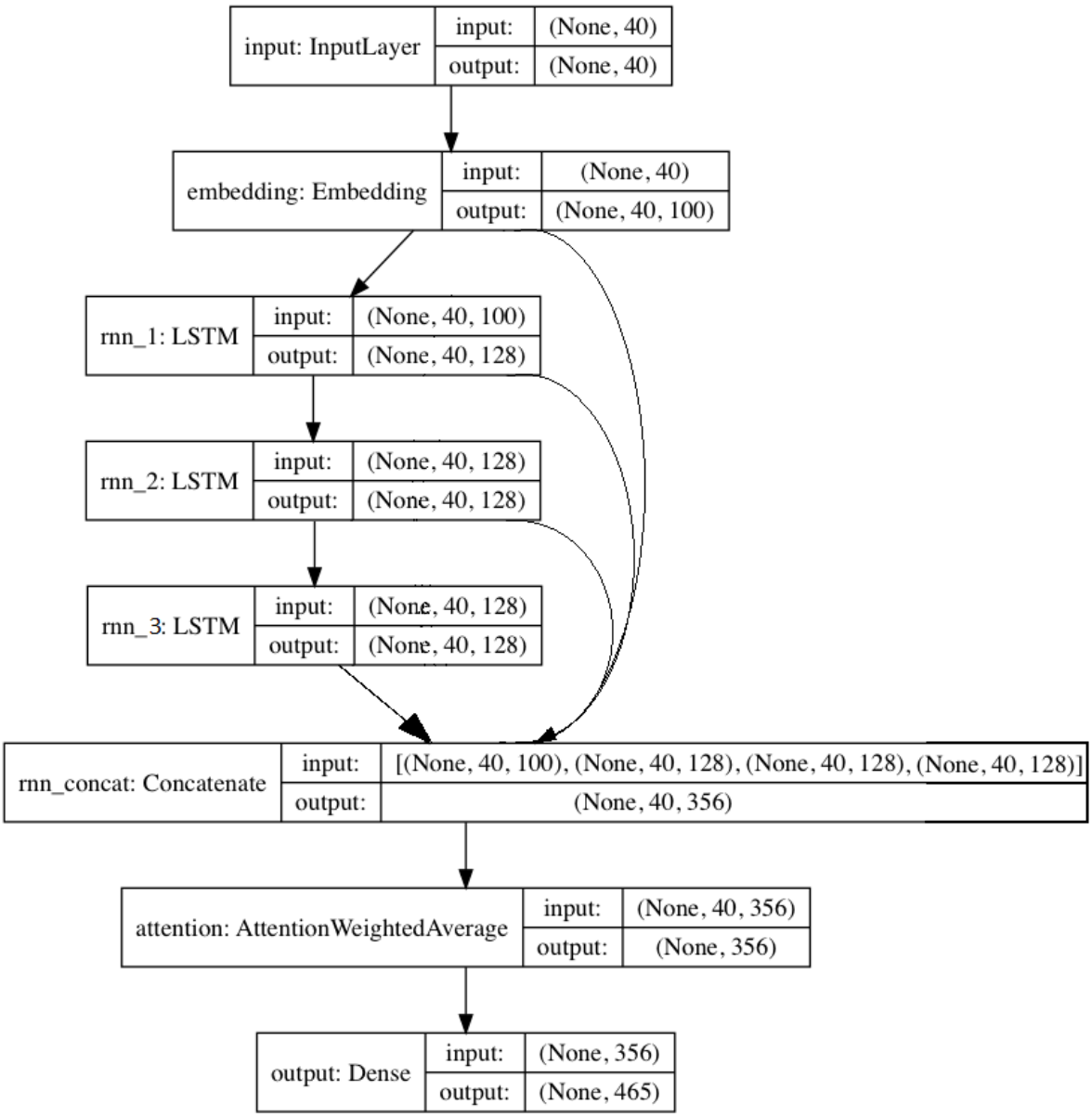

Рис. 6. Примерная архитектура модели нейронной сети

внимания, представляет собой высокоуровневое кодирование всего текста, которое используется в качестве входных данных для конечного полносвязного уровня с активацией Softmax для классификации [13]. Для проверки насколько хорошо или плохо модель обучилась за конкретную эпоху, вычисляется функция потерь Categorical Cross-Entropy, которая вычисляется 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.

Нами проведены эксперименты с изменением количества слоев LSTM (GRU) в модели (2 и 3 слоя), а также добавлением после слоя Embedding слоя Dropout, который случайным образом исключает заданное количество нейронов для предотвращения переобучения сети и лучшего обобщения модели. Сети с 3 рекуррентными слоями и Dropout показали лучший результат.

Также обучены двунаправленные модели этих сетей. Двунаправленные рекуррентные сети (Bidirectional Recurrent Neural Networks) – модель, предложенная в 1997 году Майком Шустером и Кулдип Паливал [18], которая позволяет учитывать контекст слова не только слева от него, но и справа в последовательности. Общий вид двунаправленных нейронных сетей изображен на рис. 7.

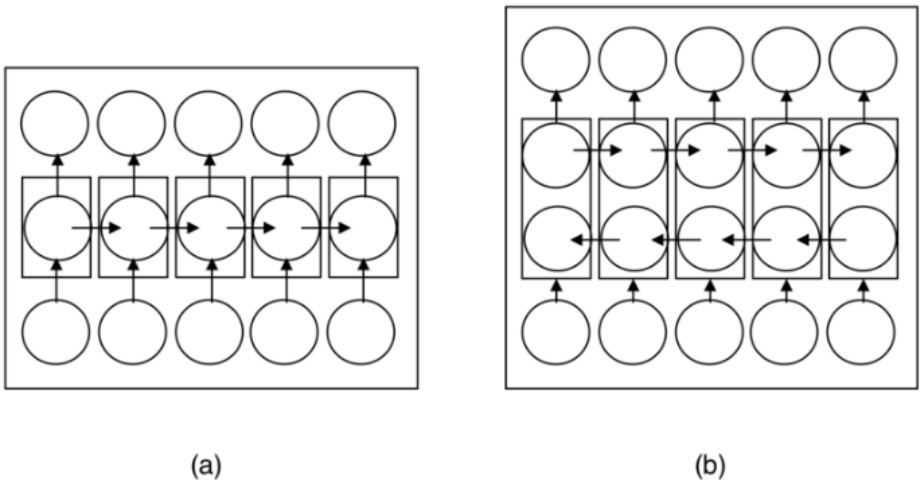

Рис. 7. Общий вид: a) однонаправленной нейронной сети; b) двунаправленной нейронной сети

В случае задачи генерации поисковых запросов двунаправленная модель показала себя лучше однонаправленной, полученные значения функции потерь после обучения моделей в течение 30 эпох указаны в таб. 1.

Таб. 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 менее значительное, что говорит о том, что двунаправленная модель в данной задаче обучается не так хорошо, а скорее «запоминает» полученные последовательности символов.

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

Трансформер

Transformer – это модель глубокого машинного обучения, представленная в 2017 году [19]. Общий вид архитектуры Transformer представлен на рис. 8.

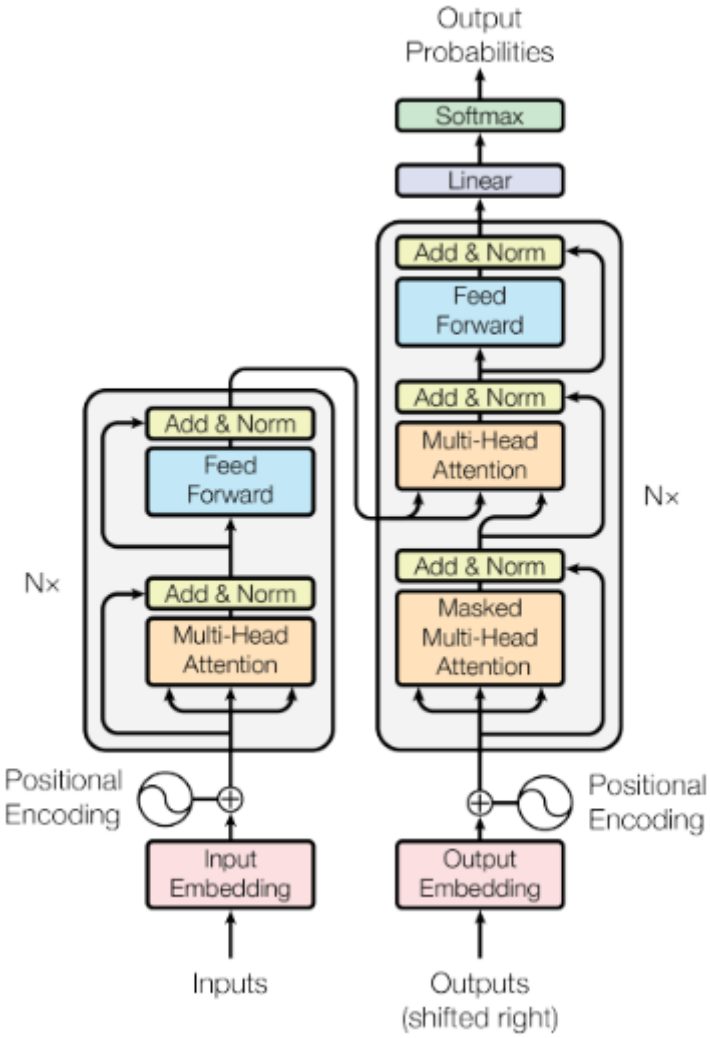

Рис. 8. Архитектура модели Transformer

Трансформеры состоят из стэков равного количества энкодеров и декодеров. Энкодеры обрабатывают последовательности, поступающие на вход, 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 во время кодирования конкретного слова. Выход этого слоя отправляется 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.

После всех декодеров используется полносвязный слой Softmax, который преобразует полученные значения в вероятности, из которых затем выбирается наибольшее значение, и слово, соответствующее ему, 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 миллиардов, 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, которые не встречаются в этих словах [23]. Этот способ обеспечивает баланс между представлениями на уровне символов и слов, что позволяет ему справляться с большими корпусами данных.

Внутреннее внимание в GPT-2 также использует маскирование, блокируя информацию от токенов справа от позиции, которая вычисляется.

Реализация GPT-2.

В работе использована модель GPT-2 среднего размера с 345 миллионами параметров, состоящая из 24 блоков декодеров.

Модель 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 сгенерированы поисковые запросы.

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

Латентно-семантический анализ.

Латентный семантический анализ (LSA) – это метод обработки естественного языка для анализа отношений между наборами документов и содержащимися в них терминами [24].

Метод использует терм-документную матрицу, которая описывает частоту вхождений 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 матрицы 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 использована библиотека gensim для Python [26]. 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 genism 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. Сгенерированные с помощью нейронных сетей запросы 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 таб. 2.

Таб. 2. Результаты латентно-семантического анализа

	GRU	LSTM	Fine-tuned GPT-2
Среднее значение	0,0065	0,006	0,0035
Среднее кол-во значений больше 0,7	16	14	9
Среднее кол-во значений больше 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 искусственно созданного с помощью сетей GRU и LSTM запроса 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 другом корпусе, она была только дообучена с помощью метода fine-tuning для того, 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 и 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 дословно, то есть являются новыми по смыслу запросами.

Сети GRU и LSTM обучены посимвольно и могли сгенерировать несуществующие слова, поэтому было решено проверить их. Нами найден корпус, содержащий более 466 тысяч английских слов, доступный по

ссылке https://github.com/dwyl/english-words, с его помощью 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 в словах, 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 тестирования 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, проведен экспертный (ручной) анализ для оценки качества сгенерированных поисковых запросов.

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

Результаты анализа приведены в таб. 3.

Таб. 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 показала себя хуже в автоматическом анализе, но лучше при экспертной оценке. Следовательно, эта модель подходит лучше для генерации поисковых запросов, так как значимость экспертной оценки выше автоматической, хотя для более точных результатов стоит провести эту оценку с помощью других экспертов.

Заключение

В ходе работы исследованы лидирующие 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 так же 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 так же проведен экспертный анализ полученных 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.

Дальнейшее развитие исследований будет направлено на использование генеративных состязательных сетей для оценки релевантности результатов поисковой системы [27].

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований, проект 18-07-00964.

Литература

1. Van Deemter, Kees. Real vs. template-based natural language generation: a false opposition? / Kees van Deemter, Emiel Krahmer, Mariet Theune, 2005. https://wwwhome.ewi.utwente.nl/~theune/PUBS/templates-squib.pdf
2. Xie Ziang. Neural Text Generation: A Practical Guide, 2017. https://arxiv.org/pdf/1711.09534.pdf
3. A Comprehensive Guide to Natural Language Generation, 2019. https://medium.com/sciforce/a-comprehensive-guide-to-natural-language-generation-dd63a4b6e548
4. Arrington, Michael. AOL proudly releases massive amounts of user search data, 2006. https://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/
5. Reiter, Ehud. NLG vs Templates: Levels of Sophistication in Generating Text, 2016. https://ehudreiter.com/2016/12/18/nlg-vs-templates
6. Gagniuc, Paul. Markov Chains: From Theory to Implementation and Experimentation, 2017. – USA, NJ: John Wiley & Sons.

7. Press, Ofir. Language Generation with Recurrent Generative Adversarial Networks without Pre-training / Ofir Press, Amir Bar, Ben Bogin, Jonathan Berant, Lior Wolf, 2017. https://arxiv.org/pdf/1706.01399.pdf
8. Williams, Ronald J. Learning representations by back-propagating errors. / Ronald J. Williams, Geoffrey E. Hinton, David E. Rumelhart, 1986. http://www.cs.utoronto.ca/~hinton/absps/naturebp.pdf
9. Hochreiter, Sepp. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies. / Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, J¨urgen Schmidhuber, 2001. https://www.bioinf.jku.at/publications/older/ch7.pdf
10. Hochreiter, Sepp. Long-Short Term Memory. / Sepp Hochreiter, Jurgen Schmidhuber, 1997. http://web.archive.org/web/20150526132154/http://deeplearning.cs.cmu.edu/pdfs/Hochreiter97_lstm.pdf
11. Heck, Joel. Simplified Minimal Gated Unit Variations for Recurrent Neural Networks / Joel Heck, Fathi M. Salem, 2017. https://arxiv.org/abs/1701.03452
12. Bahdanau, Dzmitry. Neural Machine Translation by Jointly Learning to Align and Translate. / Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio, 2016. https://arxiv.org/pdf/1409.0473.pdf
13. Felbo, Bjarke. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. / Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, Sune Lehmann, 2017. https://arxiv.org/pdf/1708.00524.pdf
14. Bisong, Ekaba. Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform, 2019. Apress, Berkeley, CA.
15. Chollet, Francois. Keras, 2015. https://keras.io
16. Kingma, Diederik. Adam: A Method for Stochastic Optimization. / Diederik P. Kingma, Jimmy Ba, 2014. https://arxiv.org/abs/1412.6980
17. Learning Rate Scheduler. https://keras.io/api/callbacks/learning_rate_scheduler/
18. Schuster, Mike. Bidirectional recurrent neural networks. / Mike Schuster, Kuldip K. Paliwal, 1997. https://www.researchgate.net/publication/3316656_Bidirectional_recurrent_neural_networks
19. Vaswani, Ashish. Attention Is All You Need. / Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, 2017. https://arxiv.org/pdf/1706.03762.pdf
20. Radford, Alec. Language Models Are Unsupervised Multitask Learners. / Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, 2018. https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf

21. Devlin, Jacob. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. / Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, 2018. https://arxiv.org/pdf/1810.04805.pdf
22. Brown, Tom B. Language Models Are Few-Shot Learners. / Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, 2019. https://arxiv.org/abs/2005.14165
23. Gage, Philip. A New Algorithm for Data Compression, 1994. https://www.derczynski.com/papers/archive/BPE_Gage.pdf
14. Deerwester, Scott. Indexing by Latent Semantic Analysis. / Scott Deerwester, Richard Harshman, 1987. https://www.cs.bham.ac.uk/~pxt/IDA/lsa_ind.pdf
25. Nakov, Preslav. Getting Better Results With Latent Semantic Indexing, 2009. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.6406&rep=rep1&type=pdf
26. Rehurek, Radim. Software Framework for Topic Modelling with Large Corpora. / Radim Rehurek, Petr Sojka, 2010. Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. University of Malta.
27. Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. Generative adversarial nets. Proceedings of the 27th International Conference on Neural Information Processing Systems, v. 2, ser. NIPS'14, Montreal, Canada: MIT Press, 2014, pp. 2672–2680.

References

1. Van Deemter, Kees. Real vs. template-based natural language generation: a false opposition? / Kees van Deemter, Emiel Krahmer, Mariet Theune, 2005. https://wwwhome.ewi.utwente.nl/~theune/PUBS/templates-squib.pdf
2. Xie Ziang. Neural Text Generation: A Practical Guide, 2017. https://arxiv.org/pdf/1711.09534.pdf
3. A Comprehensive Guide to Natural Language Generation, 2019. https://medium.com/sciforce/a-comprehensive-guide-to-natural-language-generation-dd63a4b6e548
4. Arrington, Michael. AOL proudly releases massive amounts of user search data, 2006. https://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/
5. Reiter, Ehud. NLG vs Templates: Levels of Sophistication in Generating Text, 2016. https://ehudreiter.com/2016/12/18/nlg-vs-templates
6. Gagniuc, Paul. Markov Chains: From Theory to Implementation and Experimentation, 2017. – USA, NJ: John Wiley & Sons.
7. Press, Ofir. Language Generation with Recurrent Generative Adversarial Networks without Pre-training / Ofir Press, Amir Bar, Ben Bogin, Jonathan Berant, Lior Wolf, 2017. https://arxiv.org/pdf/1706.01399.pdf

8. Williams, Ronald J. Learning representations by back-propagating errors. / Ronald J. Williams, Geoffrey E. Hinton, David E. Rumelhart, 1986. http://www.cs.utoronto.ca/~hinton/absps/naturebp.pdf
9. Hochreiter, Sepp. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies. / Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, J¨urgen Schmidhuber, 2001. https://www.bioinf.jku.at/publications/older/ch7.pdf
10. Hochreiter, Sepp. Long-Short Term Memory. / Sepp Hochreiter, Jurgen Schmidhuber, 1997. http://web.archive.org/web/20150526132154/http://deeplearning.cs.cmu.edu/pdfs/Hochreiter97_lstm.pdf
11. Heck, Joel. Simplified Minimal Gated Unit Variations for Recurrent Neural Networks / Joel Heck, Fathi M. Salem, 2017. https://arxiv.org/abs/1701.03452
12. Bahdanau, Dzmitry. Neural Machine Translation by Jointly Learning to Align and Translate. / Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio, 2016. https://arxiv.org/pdf/1409.0473.pdf
13. Felbo, Bjarke. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. / Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, Sune Lehmann, 2017. https://arxiv.org/pdf/1708.00524.pdf
14. Bisong, Ekaba. Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform, 2019. Apress, Berkeley, CA.
15. Chollet, Francois. Keras, 2015. https://keras.io
16. Kingma, Diederik. Adam: A Method for Stochastic Optimization. / Diederik P. Kingma, Jimmy Ba, 2014. https://arxiv.org/abs/1412.6980
17. Learning Rate Scheduler. https://keras.io/api/callbacks/learning_rate_scheduler/
18. Schuster, Mike. Bidirectional recurrent neural networks. / Mike Schuster, Kuldip K. Paliwal, 1997. https://www.researchgate.net/publication/3316656_Bidirectional_recurrent_neural_networks
19. Vaswani, Ashish. Attention Is All You Need. / Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, 2017. https://arxiv.org/pdf/1706.03762.pdf
20. Radford, Alec. Language Models Are Unsupervised Multitask Learners. / Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, 2018. https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf
21. Devlin, Jacob. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. / Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, 2018. https://arxiv.org/pdf/1810.04805.pdf

22. Brown, Tom B. Language Models Are Few-Shot Learners. / Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, 2019. https://arxiv.org/abs/2005.14165
23. Gage, Philip. A New Algorithm for Data Compression, 1994. https://www.derczynski.com/papers/archive/BPE_Gage.pdf
14. Deerwester, Scott. Indexing by Latent Semantic Analysis. / Scott Deerwester, Richard Harshman, 1987. https://www.cs.bham.ac.uk/~pxt/IDA/lsa_ind.pdf
25. Nakov, Preslav. Getting Better Results With Latent Semantic Indexing, 2009. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.6406&rep=rep1&type=pdf
26. Rehurek, Radim. Software Framework for Topic Modelling with Large Corpora. / Radim Rehurek, Petr Sojka, 2010. Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. University of Malta.
27. Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. Generative adversarial nets. Proceedings of the 27th International Conference on Neural Information Processing Systems, v. 2, ser. NIPS'14, Montreal, Canada: MIT Press, 2014, pp. 2672–2680.