УДК 004.912 **DOI** 10.15622/ia.21.3.3

Д.Т. Галеев, В.С. Панищев

ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ ЯЗЫКОВЫХ МОДЕЛЕЙ "ТРАНСФОРМЕР" В ЗАДАЧЕ НАХОЖДЕНИЯ ОТВЕТА НА ВОПРОС В РУССКОЯЗЫЧНОМ ТЕКСТЕ

Галеев Д.Т., Панищев В.С. **Экспериментальное исследование языковых моделей "трансформер" в задаче нахождения ответа на вопрос в русскоязычном тексте.**

Аннотация. Целью исследования является получение более легковесной языковой модели, которая сравнима по показателям EM и F-меры с лучшими современными языковыми моделям в задаче нахождения ответа на вопрос в тексте на русском языке. Результаты работы могут найти применение в различных вопросно-ответных системах, для которых важно время отклика. Поскольку более легковесная модель имеет меньшее количество параметров чем оригинальная, она может быть использована на менее мощных вычислительных устройствах, в том числе и на мобильных устройствах. В настоящей работе используются методы обработки естественного языка, машинного обучения, теории искусственных нейронных сетей. Нейронная сеть настроена и обучена с использованием библиотек машинного обучения Torch и Hugging face. В работе было проведено обучение модели DistilBERT на наборе данных SberQUAD с применением дистилляции и без. Произведено сравнение работы полученных моделей.Обученная в ходе дистилляции модель DistilBERT (EM 58,57 и F-мера 78,42) смогла опередить результаты более крупной генеративной сети ruGPT-3-medium (EM 57,60 и F-мера 77,73) притом, что ruGPT-3-medium имеет в 6,5 раз больше параметров. Также модель продемонстрировала лучшие показатели EM и F-мера, чем та же модель, но к которой применялось только обычное дообучение без дистилляции (EM 55,65, F-мера 76,51). К сожалению, полученная модель сильнее отстаёт от более крупной дискриминационной модели ruBERT (EM 66,83, F-мера 84,95), которая имеет в 3,2 раза больше параметров. Предложены направления для дальнейшего исследования.

Ключевые слова: машинное обучение, глубокое обучение, нейронные сети, обработка естественного языка, трансформер.

1. Введение. С появлением архитектуры «трансформер» вся индустрия обработки естественного языка получила значительный скачок в результатах. В работах [1-3] показано каких успехов смогли добиться различные модели данной архитектуры. Результаты «трансформера» поспособствовали появлению таких моделей как: BART [4], T5 [5], Pegasus [6], ProphetNet [7]. Также исследования показали, что использование двух отдельных частей трансформера, кодировщика и декодера, отдельно друг от друга также позволяют получить выдающийся результат. Это способствовало появлению большого числа моделей, состоящих только из кодировщика: BERT [8], ALBERT [9], RoBERTa [10], а также способствовало появлению моделей, состоящих только из декодера: GPT-3 [11], CTRL [12], Transformer-XL [13].

Одной из основных проблем использования данных моделей является большое количество параметров. Чем больше количество параметров, тем больше ресурсов необходимо затратить на обучение сети и непосредственно на её применение. Данный нюанс может быть критичен для систем, в которых важна скорость отклика. Также из-за необходимости в большой вычислительной мощности применение данных моделей может быть ограничено только мощным оборудованием, что исключает применение моделей на мобильных устройствах. Поэтому одним из направлений изучения в области обработки естественного языка является нахождение способов по уменьшению размерности моделей при сохранении качества результатов. Одним из основных способов выполнить данную задачу является дистилляция [14].

Суть дистилляции заключается в том, что можно обучить более легковесную модель, которая будет имитировать поведение более сложной модели-учителя. В качестве учителя может выступать ансамбль моделей. Модель DistilBERT показала отличные результаты по сравнению с оригинальной версией [15].

Для языковых моделей существует огромное количество задач по обработке естественного языка, среди которых: текстовый поиск, машинный перевод, написание краткого содержания текста, распознавание именованных сущностей и т.д. Также одной из основных задач для обработки естественного языка при помощи нейронных сетей является создание модели, которая ищет ответ на вопрос в тексте. Под поиском ответа на вопрос в тексте будем подразумевать: наличие текста и вопроса к тексту, система должна выбрать в качестве ответа на вопрос непрерывный фрагмент из данного текста. В англоязычной литературе данный тип задачи называется Extractive Question Answering (извлечение ответа на вопрос). Для моделей типа «кодировщик» данная задача является задачей классификации, в которой они пытаются найти ответы на вопросы: «Является ли данное слово из текста началом ответа на заданный вопрос?», «Является ли данное слово из текста концом ответа на заданный вопрос?». Для данной задачи на английском языке основным набором данных является SQuAD [16]. Для русского языка компанией Сбер был создан набор данных SberQuAD [17].

Целью исследования является получение более легковесной языковой модели, которая несильно уступает лучшим современным языковым моделям в задаче нахождения ответа на вопрос в тексте на русском языке.

2. Материалы и методы

2.1. Набор данных для обучения модели. Одним из стандартных наборов данных для вопросно-ответных систем на английском языке является Stanford Question Answering Dataset (SQuAD) [16]. Набор данных основан на статьях с сайта Wikipedia. Статьи, которые были взяты в набор, охватывают широкий спектр тем – от музыкальных знаменитостей до абстрактных понятий. Вопросы основаны на содержании статьи, и человек способен ответить на них, прочитав текст статьи. Некоторые статьи могут содержать несколько вопросов. SQuAD часто используют как бенчмарк для новых языковых моделей.

На данный момент в интернете можно найти следующие версии этого набора: SQuAD 1.1 и SQuAD 2.0.

SQuAD 1.1 содержит 107785 вопросно-ответных пар, основанных на 536 статьях. В данной версии набора присутствуют вопросы, на которые есть ответы в представленных текстах. Данный набор разбит на 3 части: тренировочную (80%), а валидационную (10%) и проверочную (10%). Средняя длина текста составляет ~755 символа (120 токенов), средняя длина вопроса составляет ~60 символа (10 токенов), средняя длина ответа составляет ~19 символов (3 токена).

SQuAD 2.0 содержит все данные из версии 1.1, но также содержит дополнительные 50 тысяч вопросов, на которые нет ответов в представленных текстах. Это бы сделано для того, чтобы модель училась понимать ситуации, когда в тексте нет необходимой информации для ответа на вопрос.

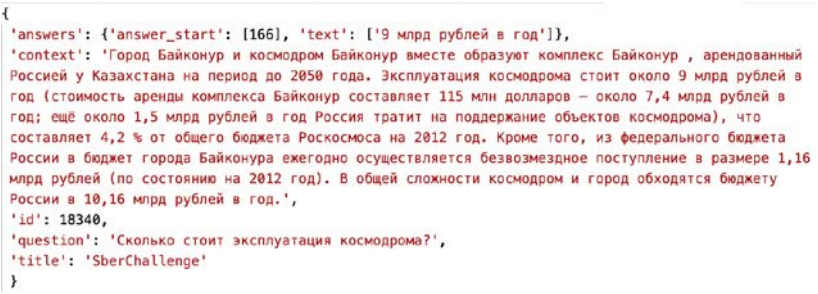
Основным набором данных для вопросно-ответных систем на русском языке является SberQuAD. Он основан на русских статьях с сайта Wikipedia. Форматом ответов и вопросов он совпадает с SQuAD 1.1. Данный набор содержит 45328 тренировочных наборов из текста, вопроса и ответа, 5036 валидационных наборов и 23936 проверочных наборов. К сожалению, ответы на проверочные данные не представлены публично, поэтому результаты работы модели сравниваются на валидационном наборе.

Большинство вопросов в наборе данных SberQuAD начинаются либо с вопросительного слова, либо с предлога. Далее представлены десять наиболее распространенных начальных слов: «что», «в», «как», «кто», «какие», «когда», «какой», «где», «сколько», «на». Средняя длина текста составляет ~754 символа (102 токена), средняя длина вопроса составляет ~64 символа (9 токенов), средняя длина ответа составляет ~26 символов (4 токена).

Конкретный экземпляр набора данных был взят из библиотеки Hugging face. Каждый экземпляр данных содержит следующие поля: context, question, answers.

В поле context находится текст, к которому будет задан вопрос, question содержит вопрос к тексту из поля contex, а answers включает в себя поле answer_start, которое содержит индекс начала ответа на вопрос, и поле text, которое содержит полный текст ответа.

Пример тренировочных данных представлен на рисунке 1.



{
'answers': {'answer_start': [166], 'text': ['9 млрд рублей в год']},
'context': 'Город Байконур и космодром Байконур вместе образуют комплекс Байконур , арендованный Россией у Казахстана на период до 2050 года. Эксплуатация космодрома стоит около 9 млрд рублей в год (стоимость аренды комплекса Байконур составляет 115 млн долларов – около 7,4 млрд рублей в год; ещё около 1,5 млрд рублей в год Россия тратит на поддержание объектов космодрома), что составляет 4,2 % от общего бюджета Роскосмоса на 2012 год. Кроме того, из федерального бюджета России в бюджет города Байконура ежегодно осуществляется безвозмездное поступление в размере 1,16 млрд рублей (по состоянию на 2012 год). В общей сложности космодром и город обходятся бюджету России в 10,16 млрд рублей в год.',
'id': 18340,
'question': 'Сколько стоит эксплуатация космодрома?',
'title': 'SberChallenge'
}

Рис. 1. Пример тренировочных данных

2.2. Выбор модели. Для опытов было решено использовать модели семейства BERT. Архитектура Bidirectional Encoder Representations from Transformers является одной из самых популярных моделей для обработки естественного языка [8], поскольку она показывает одни из лучших результатов среди аналогов. В данной модели присутствует кодировщик из архитектуры трансформер [1]. Изначально модель BERT обучалась одновременно на двух задачах:

1) предсказание пропущенных слов в тексте (masked language modeling);

2) определения является ли вторая часть текста логичным продолжением первой (next sentence prediction).

В оригинальной статье было описано две версии модели:

1) BERT BASE. Данная версия модели содержит 12 блоков трансформера. В каждом блоке трансформера содержится 12 «голов» внимания. Для каждого входного вектора на выход подаётся вектор с длиной 768. Модель содержит 110 миллионов параметров.

2) BERT LARGE. Данная версия модели содержит 24 блока трансформера. В каждом блоке трансформера содержится 16 «голов» внимания. Для каждого входного вектора на выход подаётся вектор с длиной 1024. Модель содержит 340 миллионов параметров.

Особенностью современных языковых моделей является то, что они обычно выкладываются в сеть после тренировки на различных задачах языкового моделирования или близкие к ним. Данное обучение происходит на огромных корпусах текстов. Это значит, что данные сети уже содержат векторные представления для слов из своего словаря и способны хорошо решать задачи языкового моделирования. И последующая работа с данными моделями заключается в добавлении поверх сети дополнительного полносвязного нейронного слоя с необходимой функцией активации. После этого происходит процесс дообучения модели под конкретную задачу.

Основные этапы работы модели BERT:

1) Токенизация входного текста. Добавление первым элементов специального токена [CLS].

2) Векторизация полученных токенов.

3) Применение позиционного кодирования для полученных векторов.

4) Передача векторов на вход стека блоков трансформера.

5) Для каждого входного вектора на выход подается результирующий вектор (размерность которого равна 768 в базовой модели BERT) и, в зависимости от задачи, различные результирующие вектора подаются на добавленный «сверху» слой нейронов.

Стоит отметить, что версия DistilBERT имеет ту же общую архитектуру, что и BERT, но в которой произведены некоторые упрощения, а количество слоев было уменьшено в 2 раза [15].

2.3. Токенизация. Одним из современных алгоритмов для токенизации текстов на естественном языке является Byte Pair Encoding (BPE) [18]. Основными шагами BPE являются:

1. создание словаря из всех символов языка;
2. представление слов из текста как списка символов;
3. подсчёт количества вхождений каждой пары символов;
4. объединение самых частотных пар в токен и добавление данного токена в словарь;
5. повторение пункта 4 до тех пор, пока не будет получен словарь заданного размера.

У данного алгоритма есть варианты с различными улучшениями, например BPE-Dropout [19].

Также одним из популярных алгоритмов токенизации является WordPiece [20]. WordPiece устроен похожим образом, так же как и BPE, только объединяются не самые частотные пары токенов, а

максимизирующие правдоподобие униграммной языковой модели. Именно данный алгоритм используется в модели BERT.

Применение алгоритмов таких как BPE и WordPiece позволяет бороться с проблемой отсутствия слова в словаре (out of vocabulary). Данная проблема случается, когда модель не знает входящего слова, поскольку его не было в её словаре на этапе обучения. Теперь минимальной частью слова является символ и все символы добавляются в словарь, а это значит, что любое слово сможет быть разбито на последовательность токенов, которые знает модель.

2.4. Векторизация. После того как текст был токенизирован, необходимо представить полученные токены в формат наиболее понятный для модели. Чаще всего эта задача решается сопоставлением каждого токена с вектором.

Одним из самых популярных способов для векторизации токенов является word2vec [21]. Word2vec — представляет собой малослойную искусственную нейронную сеть состоящую из двух слоев, которая обрабатывает текст, преобразуя его в «векторизованные» представления. Входными данными для данной сети являются большие корпуса текстов, из которых на выходе получается пространство векторов, размерность которых обычно не превышает несколько сотен (нет проблемы с большой размерностью векторов), где каждый токен в корпусе представлен вектором из сгенерированного пространства. Данные векторы учитывают семантическую близость слов (нет основной проблемы one-hot векторов). Данный способ может быть применён к множеству различных языков в различных задачах [22, 23].

Данная модель обычно обучается выполнять задачу языкового моделирования, т.е. модель пытается угадать одно или несколько слов, пропущенных в тексте. Для word2vec характерно 2 подхода обучения (рисунок 2):

1) Continuous Bag of Words (CBOW) — это метод, в котором модель пытается предсказать целевое слово по словам вокруг него. CBOW обычно хорошо работает на небольших наборах данных.

2) Skip-gram — это метод, в котором модель пытается предсказать слова вокруг данного целевого слова, что в точности противоположно CBOW. Skip-gram лучше работает на больших наборах данных.

Рис. 2. Методы CBOW и Skip-gram для обучения векторов токенов word2vec

Также для улучшения показателей векторов применяется негативный отбор (negative sampling). Данный подход подразумевает включать в процесс обучения пары слов, которые точно не являются соседями. Иначе модель в процессе обучения будет видеть только слова, которые являются соседями.

Также стоит отметить, что модель BERT обучает контекстно-зависимые представления. Это значит, что в зависимости от слов вокруг, слову будет сопоставляться нужный вектор. Данный подход позволяет сопоставлять омонимы с различными векторными представлениями.

2.5. Блок трансформера. Каждый блок трансформера состоит из следующих последовательных слоёв (рисунок 3):

1) слой многоголового самовнимания (multi-head self-attention);
2) слой нормализации;
3) слой прямого распространения;
4) слой нормализации.

Все блоки трансформера идентичны по структуре, но имеют разные веса.

Изначально на вход блока трансформера идут все вектора токенов предложения. Сам блок трансформера выдаёт точно такое же количество векторов, которое он получил на вход. Следовательно, все слои внутри блока возвращают то же самое количество векторов, которое они получили на входе.

Рис. 3. Структура блока трансформера

Слой прямого распространения необходим для выявления нелинейных зависимостей. Слой многоголового самовнимания использует только линейные функции. Нет смысла передавать результат от одной линейной функции к другой по цепочке, так как добавление новых линейных слоев не позволит выявить нелинейные зависимости в данных. Эта проблема решается добавлением слоя с нелинейными функциями активации между линейными слоями внимания. Обычно применяется функция активации ReLU (1).

$$ReLU(x) = max(0,x). \quad (1)$$

Само преобразование из слоя прямого распространения применяется отдельно к каждому входному вектору.

На каждый слой нормализации [24] подаётся сумма векторов из результата предыдущего слоя (т.е. слоя многоголового самовнимания или прямого распространения) с вектором, который был до преобразований из предыдущего слоя. Применение слоя нормализации необходимо для уменьшения времени обучения модели, а суммирование векторов до и после слоя с преобразованиями (skip-connection) необходимо для того, чтобы бороться с проблемой затухающих градиентов.

2.6. Слой самовнимания. Слой самовнимания кодирует взаимосвязь каждого слова с каждым другим словом в том же самом предложении, сосредотачивая большее внимание на самых значимых словах. А поскольку уделяется внимание предложением самому себе то механизм называется самовниманием. На рисунке 4 показано, как

часть механизма внимания для слова «он» фокусируется на слове «дядя» в предложении «Мой дядя самых честных правил, когда не в шутку занемог, он уважать себя заставил и лучше выдумать не мог».

Вначале слой самовнимания создаёт три матрицы для входного предложения: матрицу запросов (query), матрицу ключей (key) и матрицу значений (value). Эти матрицы создаются с помощью перемножения векторов токенов на три матрицы, которые были обучены во время процесса обучения нейронной сети. Для того чтобы получить выходной результат для данной «головы» самовнимания необходимо провести следующие преобразования (2) над полученными матрицами (рисунок 5):

$$Attention(query, key, value) = softmax\left(\frac{query \cdot key^T}{\sqrt{d_{key}}}\right) \cdot value, \quad (2)$$

где d_{key} – размерность вектора key.

Рис. 4. Демонстрация работы слоя самовнимания (программа для визуализации взята в [25])

Рис. 5. Механизм самовнимания

Большое распространение получила техника использования нескольких параллельных блоков самовнимания называемая многоголовым самовниманием (рисунок 6). Каждая «голова» представляет собой отдельный экземпляр механизма самовнимания (рисунок 5). Применение множества «голов» улучшает производительность слоя самовнимания за счет того, что разные большее количество «голов» позволяет устанавливать больше связей между токенами в предложении.

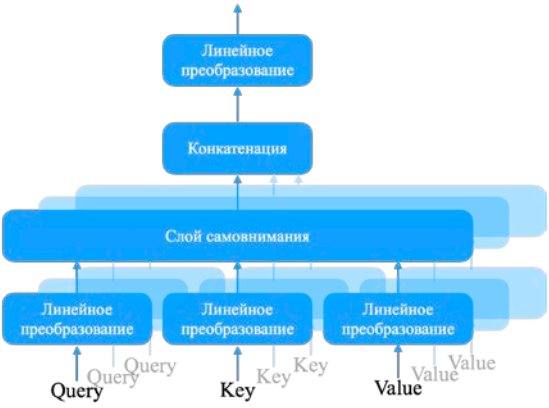


Рис. 6. Техника многоголового самовнимания

Результаты работы нескольких «голов» внимания конкатенируются друг с другом и затем умножаются на матрицу весов, которая была обучена вместе с моделью. Полученный результат передаётся в слой нормализации.

2.7. Позиционное кодирование. У применения механизма самовнимания есть один серьёзный недостаток – данный механизм не учитывает порядок слов в предложении. Часто в естественных языках порядок слов в предложении важен и не может быть отброшен или изменён. Авторы архитектуры трансформер предложили использовать «позиционное кодирование» для решения этой проблемы. Для этого добавляют специальный вектор в каждый входящий вектор токена. Эти векторы имеют определенный шаблон, который запоминает модель и который помогает определить позицию каждого слова или расстояние между разными словами в предложении. Расчёт вектора происходит по следующим формулам (3, 4):

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right), \quad (3)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right), \quad (4)$$

где, pos – это позиция токена, i – это измерение в векторе токена, d_{model} – это длина вектора токена.

2.8. Описание подхода к поиску ответа на вопрос в тексте с использованием языковых моделей на базе BERT. Под поиском ответа на вопрос в тексте будем подразумевать: наличие текста и вопроса к тексту, модель должна выбрать в качестве ответа на данный вопрос непрерывный фрагмент из данного текста. Так как в постановке задачи ответ является непрерывным отрывком из текста, то он может быть однозначно задан позициями начала и конца. Получается, что модель должна ответить на два вопроса (5) для каждого вектора токена: «Является ли данный токен из текста началом ответа на заданный вопрос?», «Является ли данный токен из текста концом ответа на заданный вопрос?» (рисунок 7):

$$Model\left(\theta, w_{start}, w_{end}\right) = -\sum_{k}\left(\log p_{start}\left(y_k^{start} | C, Q; \theta, w_{start}\right) + \log p_{end}\left(y_k^{end} | C, Q; \theta, w_{end}\right)\right), \quad (5)$$

где C - текст, Q - вопрос, Θ - параметры языковой модели, w_{start} и w_{end} - обучаемые параметры для предсказания позиции начала и конца ответа на вопрос, y_k^{start} и y_k^{end} - позиции начала и конца ответа на вопрос для примера с индексом k. Вероятности p_{start} и p_{end} при использовании модели BERT определяются следующим образом (6, 7):

$$p^{start}(y) = softmax\left(\left[\left\{w_{start}, BERT_0^N\right\}, \left\{w_{start}, BERT_1^N\right\}, \ldots, \left\{w_{start}, BERT_L^N\right\}\right]\right), \quad (6)$$

$$p^{end}(y) = softmax\left(\left[\left\{w_{end}, BERT_0^N\right\}, \left\{w_{end}, BERT_1^N\right\}, \ldots, \left\{w_{end}, BERT_L^N\right\}\right]\right), \quad (7)$$

где softmax – функция софтмакс, L — число токенов во входной последовательности, $\{\cdot,\cdot\}$ — скалярное произведение, w_{start} и w_{end} - обучаемые параметры для предсказания позиции начала и конца ответа на вопрос, $BERT_i^N$ — выход с последнего слоя N для i-ого токена во входной последовательности.

Вопрос и текст перед передачей в BERT разделяются специальным токеном [SEP].

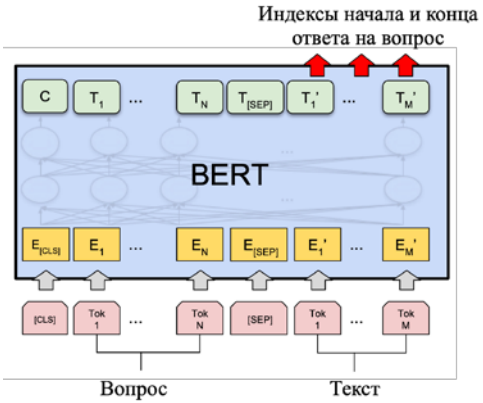


Рис. 7. Использование модели BERT для нахождения ответа на вопрос в тексте (рисунок взят из работы [8])

2.9. Дистилляция. Несмотря на то, что современные модели показывают внушительные результаты в основных задачах обработки естественного языка, данные модели часто имеют один недостаток – свой размер. Большое количество параметров модели может сильно ограничить пропускную способность конвейера, на котором используется данная модель, кроме того, большая размерность не позволяет эффективно использовать данные модели на мобильных устройствах. Поэтому одним из важных направлений исследований в области естественного языка является исследование подходов к уменьшению размера модели при сохранении той же точности работы. Одним из набирающих популярность способов уменьшения размерности моделей является дистилляция знаний.

Дистилляция знаний заключается в том, что вначале обучается большая модель, затем на основе предсказаний данной модели-учителя обучается более легковесная модель-ученик [14]. Данный подход показал хорошие результаты (полученные в ходе дистилляции модели могут совсем незначительно уступать в результатах), из-за чего появилось большое количество дистиллированных версий популярных сетей (таких как BERT, GPT). Данные дистиллированные версии содержат упрощения архитектур и меньшее количество параметров, затем данные версии моделей обучаются на основе своих больших собратьев. Также полученные модели можно дообучать под необходимую задачу.

Для задач классификации процесс дистилляции заключается в том, что модели ученика и учителя вычисляют свои распределения вероятностей по классам для конкретного экземпляра данных, а затем необходимо найти различие между этими вероятностями при помощи расхождения Кульбака-Лейблера. Полученное расхождение используется для расчета значения ошибки на текущей итерации работы сети.

3. Проведение опытов

3.1. Показатели. Основными показателями качества работы вопросно-ответных моделей являются EM и F-мера.

Exact match (EM) — точное совпадение, доля ответов системы, которые полностью совпадают с одним из правильных ответов с точностью до пунктуации и регистра.

F-мера — представляет собой совместную оценку полноты и точности. Данный показатель вычисляется по следующей формуле (8):

$$F = 2 \cdot \frac{Точность \cdot Полнота}{Точность + Полнота}. \quad (8)$$

Полнота (recall) вычисляется по следующей формуле (9):

$$Полнота = \frac{TP}{TP + FN}. \quad (9)$$

Точность (precision) вычисляется по следующей формуле (10):

$$Точность = \frac{TP}{TP + FP}, \quad (10)$$

где TP – Истинноположительные предсказания модели (модель правильно отнесла объект к классу), TN - Истинноотрицательные предсказания модели (модель правильно не отнесла объект к классу),

FN – Ложноотрицательные предсказания модели (модель неправильно не отнесла объект к классу), FP – Ложноположительные предсказания модели (модель неправильно отнесла объект к классу).

3.2. Выбор модели нейронной сети. Основной библиотекой для работы с моделями для обработки естественного языка является Hugging face. В данной библиотеке можно найти огромное количество разнообразных моделей с архитектурой «трансформер» и производными от неё. На момент написания статьи число доступных моделей в библиотеке составляет 22022. Также в данной библиотеке можно найти большое количество наборов данных. На момент написания статьи число наборов данных составляет 2059. Данная библиотека часто используется в крупных компаниях. У себя внутри библиотека
Hugging face полагается на такие библиотеки как Jax, PyTorch и TensorFlow, которые являются стандартом для всей индустрии машинного обучения. Также на сайте библиотеки в онлайне можно проверить результаты работы большинства моделей (без непосредственной установки библиотеки на персональный компьютер).

На момент написания статьи среди русскоязычных моделей для дистилляции было всего 6:

1) DeepPavlov/distilrubert-base-cased-conversational[1];
2) DeepPavlov/distilrubert-tiny-cased-conversational[2];
3) Geotrend/distilbert-base-ru-cased[3];
4) Geotrend/distilbert-base-en-ru-cased[4];
5) Geotrend/distilbert-base-en-el-ru-cased[5];
6) Geotrend/distilbert-base-en-fr-nl-ru-ar-cased[6].

Первые две не подошли к решаемой задаче, поскольку, по словам авторов, данные модели стоит использовать, только если данные решаемой задачи имеют разговорную структуру или содержат неформальный язык. Модели с 3 по 6 были основаны на мультиязычной модели distilbert-base-multilingual-cased (содержит 135×10^6 параметров), которая в свою очередь представляет дистиллированную версию bert-base-multilingual-cased (содержит 177×10^6 параметров). Данные модели были получены путём

[1] https://huggingface.co/DeepPavlov/distilrubert-base-cased-conversational
[2] https://huggingface.co/DeepPavlov/distilrubert-tiny-cased-conversational
[3] https://huggingface.co/Geotrend/distilbert-base-ru-cased
[4] https://huggingface.co/Geotrend/distilbert-base-en-ru-cased
[5] https://huggingface.co/Geotrend/distilbert-base-en-el-ru-cased
[6] https://huggingface.co/Geotrend/distilbert-base-en-fr-nl-ru-ar-cased

уменьшения поддержки 108 языков до поддержки фиксированного набора языков [26]. Было принято решение использовать модель под номером 3 поскольку она должна поддерживать только русский язык, т.е. иметь меньшее количество параметров по сравнению с остальными версиями (54×10^6 параметров в Geotrend/distilbert-base-ru-cased, против 72×10^6 в Geotrend/distilbert-base-en-ru-cased).

В качестве модели-учителя была выбрана ruBert-base поскольку среди дискриминационных моделей она показала лучшие результаты (таблица 1).

3.3. Описание стенда для обучения нейронной сети. Обучение сети проводилось на сервисе Google Colab [27]. Данный сервис предоставляет в бесплатное пользование компьютеры с производительными видеокартами (уровня NVIDIA Tesla K80) на ограниченное время (12 часов). Также доступ может быть прекращен раньше из-за простоя компьютера. Взаимодействие с компьютерами осуществляется через программу Jupyter notebook, которая поддерживает язык Python.

3.4. Обучение нейронной сети без применения процесса дистилляции. Процесс обычного обучения сети представлял собой следующие этапы:

1) Загрузку модели Geotrend/distilbert-base-ru-cased.
2) Загрузку набора данных SberQuAD.
3) Проведение тренировки модели на наборе данных со следующими параметрами обучения:
   – Шаг обучения (learning rate): 2×10^{-5};
   – Размер блока: 16;
   – Длины входных последовательностей: 384;
   – Количество эпох обучения: 3.
4) Проверка полученной модели на валидационном наборе данных.
5) Расчёт показателей EM и F-меры.
6) Время обучения модели заняло 3 часа 27 минут.

3.5. Обучение нейронной сети с применением процесса дистилляции. Обучение дистиллированной модели включало в себя следующие шаги:

1) Загрузку обученной ранее модели ruBert-base.
2) Загрузку модели Geotrend/distilbert-base-ru-cased.
3) Загрузку набора данных SberQuAD.
4) Проведение дистилляции модели Geotrend/distilbert-base-ru-cased на основе результатов, полученных из обученной ранее модели ruBert-base (2 эпохи обучения).

5) Проверка полученной модели на валидационном наборе данных.

6) Расчёт показателей EM и F-меры.

Время обучения модели заняло 4 часа 15 минут.

4. Результаты и их обсуждение. В результате обучения модели без дистилляции были получены следующие показатели на валидационном наборе данных: EM = 55,65 и F-мера = 76,51.

В результате обучения модели с дистилляцией были получены следующие показатели на валидационном наборе данных: EM = 58,57 и F-мера = 78,42.

Для сравнения были взяты модели, которые показали лучшие результаты на наборе данных SberQuAD. Данные модели представляют не только кодировщики, но еще декодеры и полные трансформеры. Стоит отметить, что для сравнения не использовались другие дистиллированные версии моделей. Результаты сравнения моделей представлены в таблице 1.

Таблица 1. Результаты сравнения моделей

Название модели	EM	F-мера	Количество параметров в модели
ruBERT-base	66,83	84,95	178×10^6
ruBERT-large	67,25	85,25	427×10^6
ruRoBERTa-large	65,23	85,45	355×10^6
ruT5-base	66,26	84,56	222×10^6
ruT5-large	68,57	86,73	737×10^6
ruGPT-3-medium	**57,60**	**77,73**	**356×10^6**
ruGPT-3-large	**59,57**	**79,51**	**760×10^6**
mT5-base (Google)	64,03	83,40	390×10^6
mT5-large (Google)	69,63	87,06	973×10^6
Модель, обученная без применения дистилляции	**55,65**	**76,51**	**54×10^6**
Модель, обученная с применением дистилляции	**58,57**	**78,42**	**54×10^6**

Модель, обученная с применением дистилляции, смогла опередить модель, которая была обучена обычным способом (EM 58,57 и F-мера 78,42 против EM 55,65 и F-мера 76,51). Также она смогла опередить генеративную модель ruGPT-3-medium (EM 58,57 и F-мера 78,42 против EM 57,60 и F-мера 77,73). Модель ruGPT-3-medium имеет 356 миллионов параметров, а модель DistilBERT имеет всего 54 миллиона параметров. Но полученная модель имеет большее отставание от более крупных

дискриминационных моделей, например ruBERT (EM 66,83, F-мера 84,95), которая имеет 178 миллионов параметров.

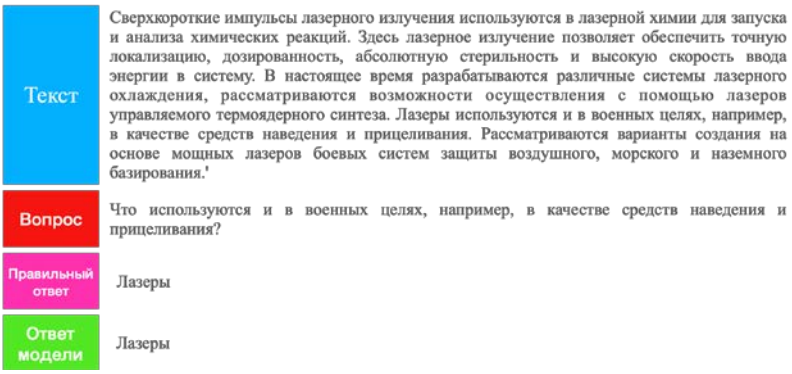


Рис. 8. Демонстрация работы модели

На рисунке 8 представлен один вопросный набор из валидационной части набора данных SberQUAD и форматированный результат работы полученной модели. Оригинальный формат ответа модели представлен на рисунке 9.

```
{
  'score': 0.870181143283844,
  'start': 435,
  'end': 441,
  'answer': 'Лазеры'
}
```

Рис. 9. Формат ответа обученной модели

5. Заключение. Обученная в ходе дистилляции модель DistilBERT (EM 58,57 и F-мера 78,42) смогла опередить результаты более крупной генеративной сети ruGPT-3-medium (EM 57,60 и F-мера 77,73) притом, что ruGPT-3-medium имеет в 6,5 раз больше параметров. К сожалению, полученная модель сильнее отстаёт от более крупной дискриминационной модели ruBERT (EM 66,83, F-мера 84,95), которая имеет в 3,2 раза больше параметров.

Учитывая большое количество готовых полных моделей следует изучить, какие результаты может показать ансамбль данных моделей [28]. Дальнейшая дистилляции полученного ансамбля может позволить получить модель, которая будет иметь значительно меньшее количество параметров и превосходить результаты моделей-учителей,

работающих по отдельности. Также можно произвести дообучение/дистилляцию на более крупных и более мелких моделях.

Литература

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. Attention is all you need // Advances in Neural Information Processing Systems 30. 2017. pp. 5998-6008.
2. Yang Z., Keung J., Yu X., Gu X., Wei Z., Ma X., Zhang M. A Multi-Modal Transformer-based Code Summarization Approach for Smart Contracts // The 2021 International Conference on Program Comprehension. 2021. pp. 1-12.
3. Juraska J., Walker M. Attention Is Indeed All You Need: Semantically Attention-Guided Decoding for Data-to-Text NLG // Proceedings of the 14th International Conference on Natural Language Generation. 2021. pp. 416-431.
4. Lewis M., Liu Y., Goyal N., Ghazvininejad M., Mohamed A., Levy O., Stoyanov V., Zettlemoyer L. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020. pp. 7871-7880.
5. Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W., Liu P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer // Journal of Machine Learning Research, Volume 21. 2020. pp .1-67.
6. Zhang J., Zhao Y., Saleh M., Liu P. J. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization // Proceedings of the 37th International Conference on Machine Learning. 2020. pp. 11328-11339.
7. Qi W., Yan Y., Gong Y., Liu D., Duan N., Chen J., Zhang R., Zhou M. ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training // Findings of the Association for Computational Linguistics: EMNLP 2020. 2020. pp. 2401–2410.
8. Devlin J., Chang M., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 2019. pp. 4171–4186.
9. Lan Z., Chen M., Goodman S., Gimpel K., Sharma P., Soricut R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. ArXiv. 2019. URL: https://arxiv.org/abs/1909.11942 (дата обращения: 12.11.2021).
10. Liu Y., Ott M., Goyal N., Du J., Joshi M., Chen D., Levy O., Lewis M., Zettlemoyer L., Stoyanov V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. ArXiv. 2019. URL: https://arxiv.org/abs/1907.11692 (дата обращения: 12.11.2021).
11. Brow T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D. M., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., McCandlish S., Radford A., Sutskever I., Amodei D. Language Models are Few-Shot Learners // Advances in Neural Information Processing Systems 33 (NeurIPS 2020). 2020. pp. 1877-1901.
12. Keskar N.S., McCann B., Varshney L. R., Xiong C., Socher R. CTRL: A Conditional Transformer Language Model for Controllable Generation. ArXiv. 2019. URL: https://arxiv.org/abs/1909.05858 (дата обращения: 12.11.2021).
13. Dai Z., Yang Z., Yang Y., Carbonell J., Le Q.V., Salakhutdinov R. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019. pp. 2978-2988.

14. Hahn S., Choi H. Self-Knowledge Distillation in Natural Language Processing // Proceedings of the International Conference on Recent Advances in Natural Language Processing, Varna, Bulgaria, September 2-4, 2019. 2019. pp. 423-430.
15. Sanh V., Debut L., Chaumond J., Wolf T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. ArXiv. 2019. URL: https://arxiv.org/abs/1910.01108 (дата обращения: 12.11.2021).
16. Rajpurkar P., Zhang J., Lopyrev K., Liang P. SQuAD: 100,000+ Questions for Machine Comprehension of Text // Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 2016. pp. 2383–2392.
17. Efimov P., Chertok A., Boytsov L., Braslavski P. SberQuAD - Russian Reading Comprehension Dataset: Description and Analysis // Experimental IR Meets Multilinguality, Multimodality, and Interaction - 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22-25, 2020, Proceedings. 2020. pp. 3-15.
18. Sennrich R., Haddow B., Birch A. Neural Machine Translation of Rare Words with Subword Units // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2016. pp. 1715–1725.
19. Provilkov I., Emelianenko D., Voita E. BPE-Dropout: Simple and Effective Subword Regularization. ArXiv. 2020 // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020. pp. 1882–1892.
20. Schuster M., Nakajima K. Japanese and Korea voice search // 2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, March 25-30, 2012. 2012. pp. 5149-5152.
21. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space. ArXiv. 2013. URL: https://arxiv.org/pdf/1301.3781.pdf (дата обращения: 12.11.2021).
22. Фат Х.Н., Ань Н.Т.М. Алгоритм классификации вьетнамского текста с использованием долгой краткосрочной памяти и Word2Vec // Информатика и автоматизация. 2020. № 6 (19). С. 1255-1279.
23. Алтаф С., Iqbal S., Soomro M.W. Эффективный алгоритм классификации естественного языка обнаружения повторяющихся контролируемых признаков // Информатика и автоматизация. 2021. № 3 (20). С. 623-653.
24. Lei Ba J., Kiros J.R., Hinton G.E. Layer Normalization. ArXiv. 2016. URL: https://arxiv.org/pdf/1607.06450.pdf (дата обращения: 12.11.2021).
25. URL: https://github.com/jessevig/bertviz (дата обращения: 12.11.2021).
26. Abdaoui A., Pradel C., Sigel G. Load What You Need: Smaller Versions of Multilingual BERT. ArXiv. 2020. URL: https://arxiv.org/abs/2010.05609 (дата обращения: 12.11.2021).
27. URL: https://colab.research.google.com/notebooks/welcome.ipynb?hl=ru (дата обращения: 12.11.2021).
28. Li S., Li R., Peng V. Ensemble ALBERT on SQuAD 2.0. ArXiv. 2021. URL: https://arxiv.org/abs/2110.09665 (дата обращения: 12.11.2021).

Галеев Денис Талгатович — аспирант, ФГБОУ ВО Юго-Западный государственный университет (ЮЗГУ). Область научных интересов: искусственный интеллект, машинное обучение, обработка естественного языка. Число научных публикаций — 15. ra3wvw@mail.ru; улица 50 лет Октября, 94, 305040, Курск, Россия; р.т.: +7(4712)222-665.

Панищев Владимир Славиевич — канд. техн. наук, доцент, кафедра вычислительной техники, ФГБОУ ВО Юго-Западный государственный университет (ЮЗГУ). Область научных интересов: нейронные сети, обработка изображений. Число научных публикаций — 150. gskunk@yandex.ru; улица 50 лет Октября, 94, 305040, Курск, Россия; р.т.: +7(4712)222626.

DOI 10.15622/ia.21.3.3

D. GALEEV, V. PANISHCHEV

EXPERIMENTAL STUDY OF LANGUAGE MODELS OF "TRANSFORMER" IN THE PROBLEM OF FINDING THE ANSWER TO A QUESTION IN A RUSSIAN-LANGUAGE TEXT

Galeev D., Panishchev V. **Experimental Study of Language Models of "Transformer" in the Problem of Finding the Answer to a Question in a Russian-Language Text.**

Abstract. The aim of the study is to obtain a more lightweight language model that is comparable in terms of EM and F1 with the best modern language models in the task of finding the answer to a question in a text in Russian. The results of the work can be used in various question-and-answer systems for which response time is important. Since the lighter model has fewer parameters than the original one, it can be used on less powerful computing devices, including mobile devices. In this paper, methods of natural language processing, machine learning, and the theory of artificial neural networks are used. The neural network is configured and trained using the Torch and Hugging face machine learning libraries. In the work, the DistilBERT model was trained on the SberQUAD dataset with and without distillation. The work of the received models is compared. The distilled DistilBERT model (EM 58,57 and F1 78,42) was able to outperform the results of the larger ruGPT-3-medium generative network (EM 57,60 and F1 77,73), despite the fact that ruGPT-3-medium had 6,5 times more parameters. The model also showed better EM and F1 metrics than the same model, but to which only conventional training without distillation was applied (EM 55,65, F1 76,51). Unfortunately, the resulting model lags further behind the larger robert discriminative model (EM 66,83, F1 84,95), which has 3,2 times more parameters. The application of the DistilBERT model in question-and-answer systems in Russian is substantiated. Directions for further research are proposed.

Keywords: machine learning, deep learning, neural networks, natural language processing, transformer.

Galeev Denis — Ph.D., Graduate student, Southwest State University (SWSU). Research interests: artificial intelligence, machine learning, natural language processing. The number of publications — 15. ra3wvw@mail.ru; 94, 50 let Oktyabrya St., 305040, Kursk, Russia; office phone: +7(4712)222-665.

Panishchev Vladimir — Ph.D., Associate professor, Department of computer engineering, Southwest State University (SWSU). Research interests: neural networks, image processing. The number of publications — 150. gskunk@yandex.ru; 94, 50 let Oktyabrya St., 305040, Kursk, Russia; office phone: +7(4712)222626.

References

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. Attention is all you need // Advances in Neural Information Processing Systems 30. 2017. pp. 5998-6008.
2. Yang Z., Keung J., Yu X., Gu X., Wei Z., Ma X., Zhang M. A Multi-Modal Transformer-based Code Summarization Approach for Smart Contracts // The 2021 International Conference on Program Comprehension. 2021. pp. 1-12.
3. Juraska J., Walker M. Attention Is Indeed All You Need: Semantically Attention-Guided Decoding for Data-to-Text NLG // Proceedings of the 14th International Conference on Natural Language Generation. 2021. pp. 416-431.

4. Lewis M., Liu Y., Goyal N., Ghazvininejad M., Mohamed A., Levy O., Stoyanov V., Zettlemoyer L. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020. pp. 7871-7880.
5. Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W., Liu P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer // Journal of Machine Learning Research, Volume 21. 2020. pp .1-67.
6. Zhang J., Zhao Y., Saleh M., Liu P. J. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization // Proceedings of the 37th International Conference on Machine Learning. 2020. pp. 11328-11339.
7. Qi W., Yan Y., Gong Y., Liu D., Duan N., Chen J., Zhang R., Zhou M. ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training // Findings of the Association for Computational Linguistics: EMNLP 2020. 2020. pp. 2401–2410.
8. Devlin J., Chang M., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 2019. pp. 4171–4186.
9. Lan Z., Chen M., Goodman S., Gimpel K., Sharma P., Soricut R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. ArXiv. 2019. URL: https://arxiv.org/abs/1909.11942 (accessed: 12.11.2021).
10. Liu Y., Ott M., Goyal N., Du J., Joshi M., Chen D., Levy O., Lewis M., Zettlemoyer L., Stoyanov V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. ArXiv. 2019. URL: https://arxiv.org/abs/1907.11692 (accessed: 12.11.2021).
11. Brow T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D. M., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., McCandlish S., Radford A., Sutskever I., Amodei D. Language Models are Few-Shot Learners // Advances in Neural Information Processing Systems 33 (NeurIPS 2020). 2020. pp. 1877-1901.
12. Keskar N.S., McCann B., Varshney L. R., Xiong C., Socher R. CTRL: A Conditional Transformer Language Model for Controllable Generation. ArXiv. 2019. URL: https://arxiv.org/abs/1909.05858 (accessed: 12.11.2021).
13. Dai Z., Yang Z., Yang Y., Carbonell J., Le Q.V., Salakhutdinov R. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019. pp. 2978-2988.
14. Hahn S., Choi H. Self-Knowledge Distillation in Natural Language Processing // Proceedings of the International Conference on Recent Advances in Natural Language Processing, Varna, Bulgaria, September 2-4, 2019. 2019. pp.423-430.
15. Sanh V., Debut L., Chaumond J., Wolf T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. ArXiv. 2019. URL: https://arxiv.org/abs/1910.01108 (accessed: 12.11.2021).
16. Rajpurkar P., Zhang J., Lopyrev K., Liang P. SQuAD: 100,000+ Questions for Machine Comprehension of Text // Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 2016. pp. 2383–2392.
17. Efimov P., Chertok A., Boytsov L., Braslavski P. SberQuAD - Russian Reading Comprehension Dataset: Description and Analysis // Experimental IR Meets Multilinguality, Multimodality, and Interaction - 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22-25, 2020, Proceedings. 2020. pp. 3-15.

18. Sennrich R., Haddow B., Birch A. Neural Machine Translation of Rare Words with Subword Units // Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2016. pp. 1715–1725.
19. Provilkov I., Emelianenko D., Voita E. BPE-Dropout: Simple and Effective Subword Regularization. ArXiv. 2020 // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020. pp. 1882–1892.
20. Schuster M., Nakajima K. Japanese and Korea voice search // 2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, March 25-30, 2012. 2012. pp. 5149-5152.
21. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations in Vector Space. ArXiv. 2013 URL: https://arxiv.org/pdf/1301.3781.pdf (accessed: 12.11.2021).
22. Phat H.N., Anh N.T.M. Vietnamese Text Classification Algorithm using Long Short Term Memory and Word2Vec // Informatics and Automation. 2020. № 6 (19). pp. 1255-1279.
23. Altaf S., Iqbal S., Soomro M.W. Efficient natural language classification algorithm for detecting duplicate unsupervised features // Informatics and Automation. 2021. № 3 (20). pp. 623-653.
24. Lei Ba J., Kiros J.R., Hinton G.E. Layer Normalization. ArXiv. 2016. URL: https://arxiv.org/pdf/1607.06450.pdf (accessed: 12.11.2021).
25. URL: https://github.com/jessevig/bertviz (accessed: 12.11.2021).
26. Abdaoui A., Pradel C., Sigel G. Load What You Need: Smaller Versions of Multilingual BERT. ArXiv. 2020. URL: https://arxiv.org/abs/2010.05609 (accessed: 12.11.2021).
27. URL: https://colab.research.google.com/notebooks/welcome.ipynb?hl=ru (accessed: 12.11.2021).
28. Li S., Li R., Peng V. Ensemble ALBERT on SQuAD 2.0. ArXiv. 2021. URL: https://arxiv.org/abs/2110.09665 (accessed: 12.11.2021).